MSX POCKET BANK 5

ポケットバンク

スプライト機能を使いこなす

# ゲームキャラクタ操縦法

横溝和宏



インベーダーも、アニメーションも
自由自在のグラフィックステクニック

アスキー出版局

MSX POCKET BANK 5

# ゲームキャラクタ操縦法

横溝和宏

アスキー出版局

CO

NTENTS

# 本書を読む前に

MSX　コンピュータには，他のマイコンにないとても魅力的な機能があります．

これはスプライト機能といって，グラフィックスに関するやや特殊なものです．

この機能はマニュアルを読んだかぎりでは，何のヘンテツもないBASICの命令の1つです．しかし，少し使い方をおぼえれば，手軽にゲームやアニメーションのプログラムが作れてしまう，便利なものです．

本書には，これを利用したゲームプログラムをいくつか載せてみました．ぜひスプライトの楽しさを目で見てください．

そして，魅力をたしかめた後，使い方をマスターしてください．アイデアしだいで，ゲームセンターのテレビゲームよりもっとおもしろいものができるでしょう．

スプライトは応用範囲の広い機能です．ゲームといわず，MSXの楽しさを引きだす可能性に満ちています．本書を読み終えた後，きみ自身のスプライト利用術を発見してください．

この本では，スプライト機能の使い方を説明しながら，マニュアルに書かれていない楽しいスプライト活用法を見ていくことにします．

昭和59年3月1日

横溝和宏

表紙デザイン　HIT STUDIO

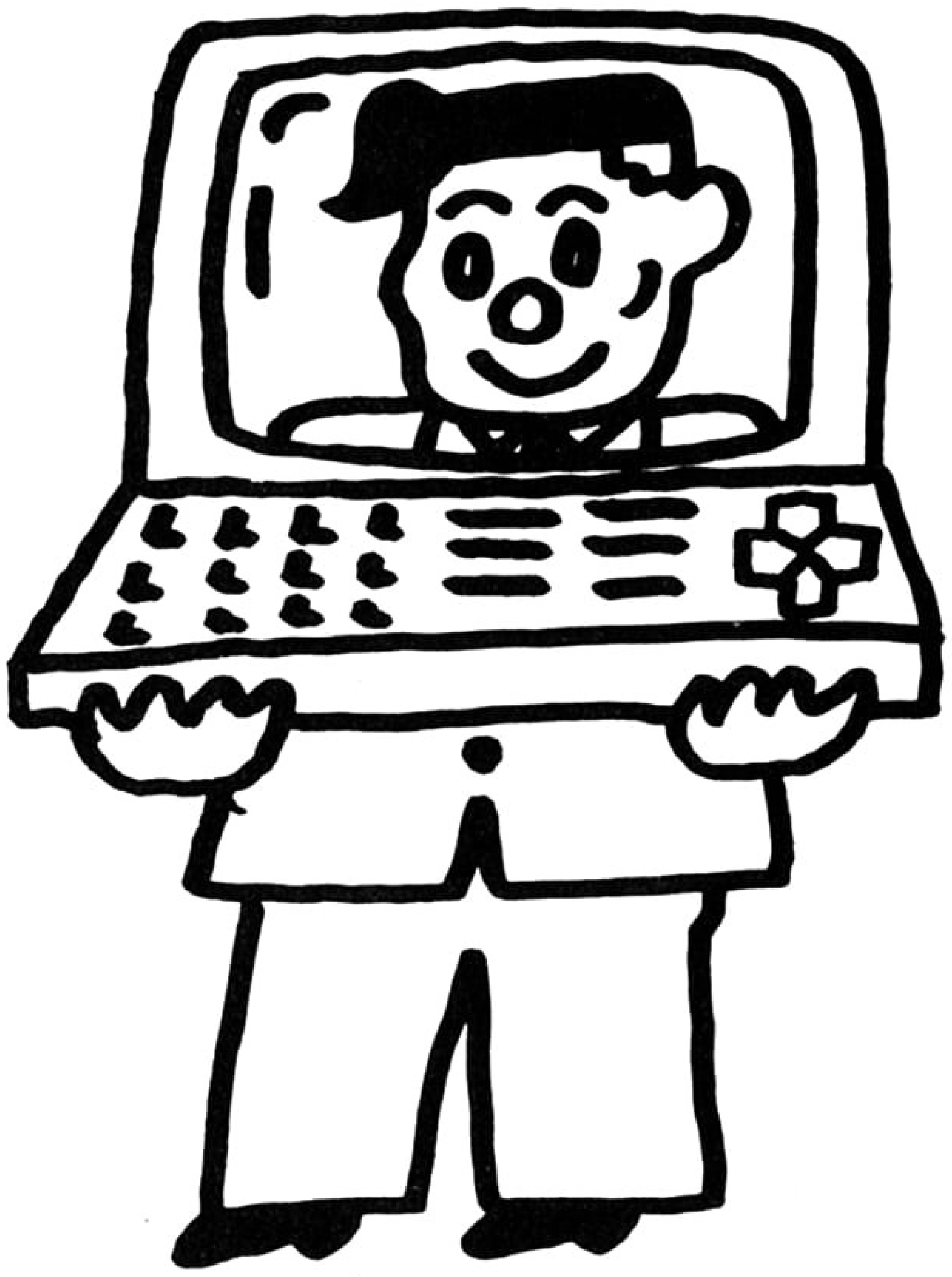
表紙イラストレーション　ナギノ タカヒロ

本文イラストレーション　藤岡 仁

GAME CHARACTER

# 1

# スプライトの実力



# 1 スプライトはゲーム作りのミソ

MSXを持っているきみたちは，もう何かのゲームで遊んでみましたか？「ゲームのために買ったんじゃないョ」という人でも，テレビゲームのおもしろさを知っているはずです．たとえば，インベーダーゲームは誰でも知っているゲームの1つですが，あのインベーダーやビーム砲の動きはどんなプログラムになっているのでしょうか．

テレビゲームもいろいろなものがあります．その中で，最もポピュラーなものは，ゲームの小さな登場人物(？)たちが画面上をすばやく動きまわるタイプでしょう．ブロックくずし，インベーダーゲーム，パックマンなど，どれも小さなゲームキャラクタがすばやく画面を走ることが基本になっています．そして，これらのゲームキャラクタはたいてい，勝手に動く敵と，操作ができる自分とにわかれ，それぞれの動きを巧みに組み合わせています．つまり，画面を走るという単純な動きを，いかに上手に組み合わせるかが，ゲームのおもしろさになっているわけです．

一方，マイコンの初心者はそうしたおもしろいゲームをなかなか作れないものです．この理由は，「ゲームキャラクタを動かす」ことが，いざプログラムにするとなると案外むずかしく，めんどうだからではないでしょうか？

そういった機能が手軽に使えれば，ゲームセンターでよく見かけるようなスリリングなゲームを作るのも簡単になり，多くの人がゲームを作ることになじめるのです．

このようなタイプのゲームを作るのに向いている機能が，この本で紹介する**スプライト(SPRITE)機能**です．ゲームキャラクタのような決まった絵のパターンをスムースに動かせ，しかもそのプログラムが非常に簡単なのが，スプライト機能の目立って便利な特徴です．

どんなものであるかは，直接目で確認するのが早道ですから，1つ例を取り上げてみましょう．

リスト1-1

```
100 COLOR 13,1,1:SCREEN 2,1:CLS
110 DEF FNX(N)=(N<10 OR N>230)
120 DEF FNY(N)=(N<0 OR N>170)
130 M=0:MX=104:MY=96:MP=-2:MQ=5
140 S=1:SX=120:SY=96:SQ=4
150 X=2:XX=136:XY=96:XP=2:XQ=-5
160 GOSUB 310
170 '
180 PUT SPRITE M,(MX,MY),2
190 PUT SPRITE S,(SX,SY),10
200 PUT SPRITE X,(XX,XY),8
210 MX=MX+MP:MY=MY+MQ
220 IF FNX(MX) THEN MX=MX-MP:MP=-1*MP
230 IF FNY(MY) THEN MY=MY-MQ:MQ=-1*MQ
240 SY=SY+SQ
250 IF FNY(SY) THEN SY=SY-SQ:SQ=-1*SQ
260 XX=XX+XP:XY=XY+XQ
270 IF FNX(XX) THEN XX=XX-XP:XP=-1*XP
280 IF FNY(XY) THEN XY=XY-XQ:XQ=-1*XQ
290 GOTO 180
300 '
310 FOR I=1 TO 8
320  READ A$,B$,C$
330  M$=M$+CHR$(VAL("&B"+A$))
340  S$=S$+CHR$(VAL("&B"+B$))
350  X$=X$+CHR$(VAL("&B"+C$))
360 NEXT
370 SPRITE$(M)=M$
380 SPRITE$(S)=S$
390 SPRITE$(X)=X$
400 RETURN
410 '
420 DATA 10000010,01111100,10000010
430 DATA 11000110,10000010,01000100
440 DATA 10101010,10000000,00101000
450 DATA 10010010,01111100,00010000
460 DATA 10010010,00000010,00010000
470 DATA 10000010,10000010,00101000
480 DATA 10000010,10000010,01000100
490 DATA 10000010,01111100,10000010
```

このプログラムは，単なるスプライト機能のデモンストレーションです．私たちは見ているだけでいいのですが，比較的多い動きでも簡単にこなしているのは，プログラムの単純さからよくわかると思います．

いままでのコンピュータですと，あるゲームキャラクタを動かすときには，「前の位置にあったキャラクタを消してから，目的の位置に描きなおす」といった，たいへんめんどうな手順が必要だったのですが，「前のキャラクタを消す」という操作が不要なスプライト機能は，見ていて本当にスムースな動きを見せてくれます．このようななめらかな動きを得ることができるのが，スプライト機能なのです．

# 2 カーソルをひと工夫

スプライト機能では，ゲームキャラクタの形を決まった大きさの中で自由に作れます．

この特徴を考えると，ゲーム以外にもいろいろ使えそうです．たとえば，気のきいたカーソルを作ってみてはどうでしょうか？

じつをいうと，スプライト機能にはもう１つ，カーソルにうってつけの特徴があるのです．それはこの機能を利用して作ったキャラクタが，画面上に表示された他の文字や絵を通り抜け，これらの表示にまったく影響を与えない，ということです．

そこで，矢印の形をしたカーソルを作っておき，画面に表示されているメニューをカーソルの移動によって自由に選べるようにプログラムを組めば，プログラム使用時にはキー操作でめんどうな思いをしなくてすみます．

また，十字形のカーソルを用意しておくと，グラフィック・カーソルとして用いることもできます．これでグラフィックの画面上の正確な位置を指定できるようにすれば，簡単に図形を描くことのできる道具(ツール)ができ上がってしまいます．

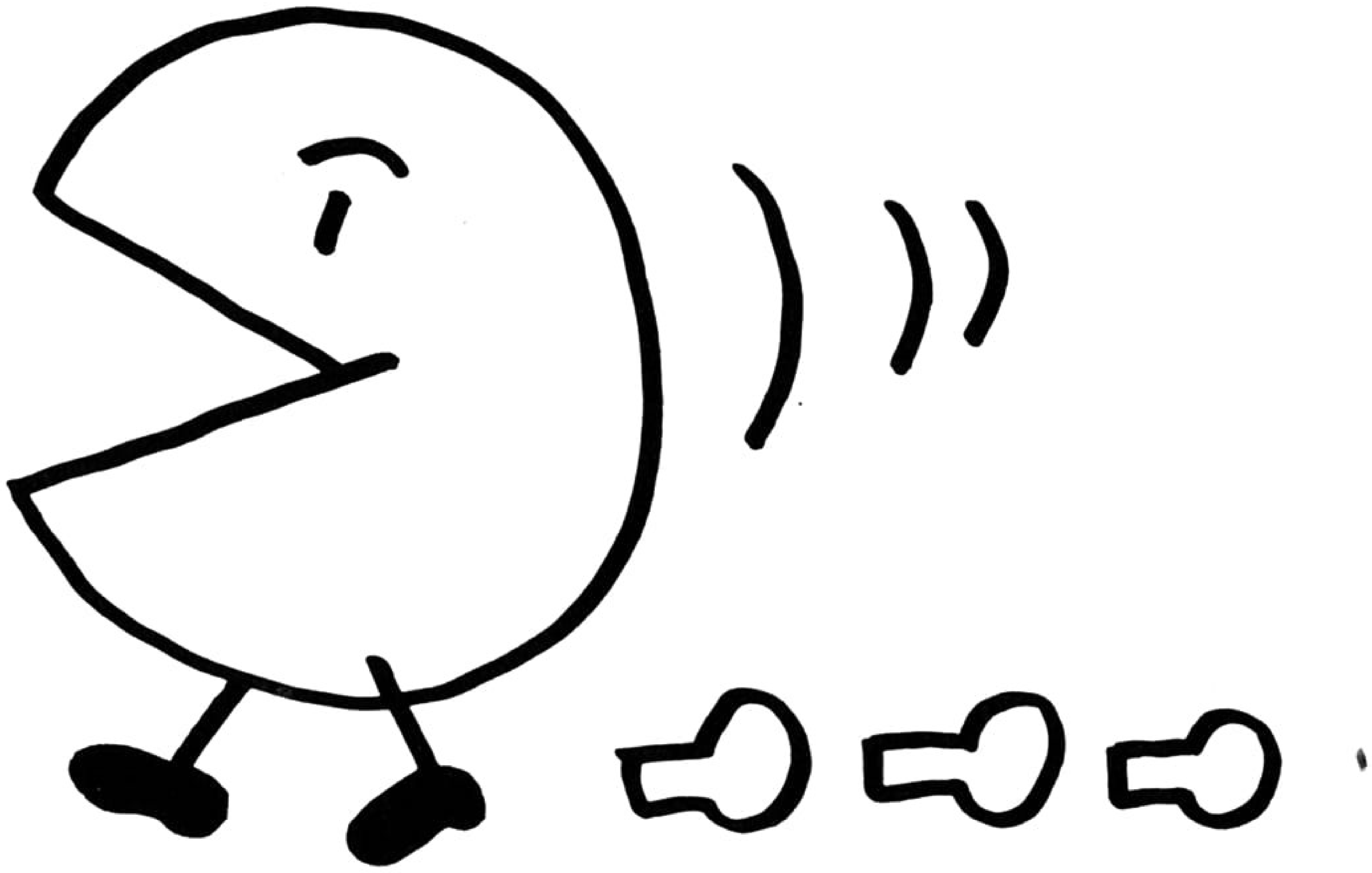
### リスト1-2

```
100 COLOR 15,4,7:SCREEN 2,1,0:CLS
110 GOSUB 370:X=200:Y=160
120 '
130 OPEN "GRP:" FOR OUTPUT AS #1
```

```
140 PRESET(32,32)
150 PRINT #1,STRING$(25,"*")
160 PRESET(56,56)
170 PRINT #1,"C  D  E  F  G  A  B"
180 PRESET(56,96)
190 PRINT #1,"Cm Dm Em Fm Gm Am Bm"
200 PRESET(32,128)
210 PRINT #1,STRING$(8,"*");">> ﾜｵﾝ  ";
220 PRINT #1,"<<";STRING$(8,"*")
230 CLOSE #1
240 '
250 PUT SPRITE 0,(X,Y),3
260 C=ASC(INPUT$(1))
270 IF C=13 OR C=27 THEN 350
280 IF C=28 AND X<250 THEN X=X+4
290 IF C=29 AND X>0   THEN X=X-4
300 IF C=30 AND Y>0   THEN Y=Y-4
310 IF C=31 AND Y<180 THEN Y=Y+4
320 IF C=32 THEN GOSUB 580
330 GOTO 250
340 '
350 END
360 '
370 DIM V(6)
380 FOR I=0 TO 6:READ V(I):NEXT
390 '
400 FOR I=1 TO 8
410  READ D$:D=VAL("&B"+D$)
420  S$=S$+CHR$(D)
430 NEXT
440 SPRITE$(0)=S$
450 RETURN
460 '
470 DATA 36,38,40,41,43,45,47
480 '
490 DATA 11110000
500 DATA 11000000
510 DATA 10100000
520 DATA 10010000
530 DATA 00001000
540 DATA 00000100
550 DATA 00000010
560 DATA 00000001
570 '
```

```
580 P=0
590 IF Y>55 AND Y<64 THEN P=4:Q=3
600 IF Y>95 AND Y<104 THEN P=3:Q=4
610 IF P=0 THEN RETURN
620 B=(X-56)/24:B=V(B)
630 A$="N"+STR$(B)
640 B$="N"+STR$(B+P)
650 C$="N"+STR$(B+P+Q)
660 PLAY "L8","L8","L8"
670 PLAY A$:PLAY B$:PLAY C$:PLAY "R"
680 PLAY "L2","L2","L2"
690 PLAY A$,B$,C$
700 RETURN
```

**リスト 1－2** に示したのは，矢印をカーソルとして使用した例です．カーソル移動キーで，画面上の矢印を表示されているメニューの好きな位置に合わせて，スペースバーを打つと，その和音が出るプログラムです．単にメニューに合わせて音を出すだけの機能しか持っていませんが，これに編集する機能などをつけたせば，かなりおもしろいプログラムになると思います．



# 3 パズルゲームをやってみよう

さて，テレビゲームの話題からはじめながらゲームらしいプログラムが1つも出てきていないので，このあたりで何か簡単な例をあげてみましょう．複雑な動きを含んだゲームは後回しにすることにして，上下左右に図形を動かすだけでゲームが楽しめるものとして，次のプログラムを考えてみました．

リスト1-3

```
1000 GOTO 1480
1010 '
1020 IF P>11 AND P<16 THEN RETURN
1030 SWAP C(P),C(P+4):O=P:P=P+4
1040 YS=INT(P/4)*16+64
1050 YD=YS-16:D=-1
1060 GOTO 1130
1070 '
1080 IF P>-1 AND P<4 THEN RETURN
1090 SWAP C(P-4),C(P):O=P:P=P-4
1100 YS=INT(P/4)*16+64
1110 YD=YS+16:D=1
1120 '
1130 X=(P MOD 4)*16+96
1140 FOR Y=YS TO YD STEP D
1150  IF F THEN 1170
1160  PUT SPRITE C(O),(X,Y)
1170 NEXT
1180 RETURN
1190 '
1200 IF (P MOD 4)=0 THEN RETURN
1210 SWAP C(P-1),C(P):O=P:P=P-1
1220 XS=(P MOD 4)*16+96
1230 XD=XS+16:D=1
1240 GOTO 1310
1250 '
1260 IF (P MOD 4)=3 THEN RETURN
1270 SWAP C(P),C(P+1):O=P:P=P+1
1280 XS=(P MOD 4)*16+96
1290 XD=XS-16:D=-1
```

```
1300 '
1310 Y=INT(P/4)*16+64
1320 FOR X=XS TO XD STEP D
1330  IF F THEN 1350
1340  PUT SPRITE C(O),(X,Y)
1350 NEXT
1360 RETURN
1370 '
1380 FOR I=0 TO 14:S$=""
1390  FOR J=1 TO 32
1400   READ D$:D=VAL("&H"+D$)
1410   S$=S$+CHR$(D)
1420  NEXT
1430  SPRITE$(I)=S$
1440 NEXT
1450 SPRITE$(15)=STRING$(32,0)
1460 RETURN
1470 '
1480 COLOR 15,4,7:SCREEN 2,2:CLS
1490 ON STRIG GOSUB 1910:STRIG(0) ON
1500 DEFINT A-Z:DIM C(15)
1510 GOSUB 1380   'MAKE PATTERN
1520 OPEN "GRP:" FOR OUTPUT AS #1
1530 PRESET(50,30)
1540 PRINT #1,"***** 15 GAME *****"
1550 PRESET(50,160)
1560 PRINT #1,"*******************"
1570 CLOSE #1
1580 FOR I=0 TO 15
1590  C(I)=I
1600  X=(I MOD 4)*16+96
1610  Y=INT(I/4)*16+64
1620  PUT SPRITE I,(X,Y)
1630 NEXT
1640 P=15:F=-1
1650 '
1660 FOR I=0 TO 100
1670  M=RND(RND(-TIME))*4+1
1680  ON M GOSUB 1020,1200,1080,1260
1690 NEXT
1700 F=NOT F
1710 FOR I=0 TO 4
1720  X=I*16+96
1730  Y=I*16+64
```

```
1740  LINE(X,64)-STEP(0,64),5
1750  LINE(96,Y)-STEP(64,0),5
1760 NEXT
1770 '
1780 FOR I=0 TO 15
1790  X=(I MOD 4)*16+96
1800  Y=INT(I/4)*16+64
1810  PUT SPRITE C(I),(X,Y)
1820 NEXT
1830 TIME=0
1840 '
1850 IF STICK(0)=1 THEN GOSUB 1020
1860 IF STICK(0)=5 THEN GOSUB 1080
1870 IF STICK(0)=3 THEN GOSUB 1200
1880 IF STICK(0)=7 THEN GOSUB 1260
1890 GOTO 1850
1900 '
1910 STRIG(0) OFF
1920 T=0
1930 FOR S=0 TO 15
1940  IF C(S)<>S THEN S=15:T=NOT T
1950 NEXT
1960 IF T THEN STRIG(0) ON:RETURN
1970 RETURN 1990
1980 '
1990 SCREEN 0:CLS
2000 PRINT "YOUR TIME IS ";
2010 PRINT USING "###.## SEC.";TIME/60
2020 PRINT:PRINT
2030 END
2040 '
2050 DATA 0,0,0,0,0,0,0,0,0,0,1,1,1,1,3,
3
2060 DATA 1F,3F,3F,7F,7F,FF,FC,F8,F0,F0,
E0,E0,C0,C0,80,80
2070 DATA FF,FF,FF,F1,C0,0,0,0,0,0,0,0,0
,0,0,0
2080 DATA FF,FF,FF,FF,3F,1F,0,0,0,0,0,0,
0,0,0,0
2090 DATA FF,FF,FF,F8,E0,C0,0,0,0,0,0,0,
0,0,0,0
2100 DATA FF,FF,FF,FF,1F,7,1,1,0,0,0,0,0
,0,0,0
2110 DATA C0,E0,E0,F0,F0,F8,F8,F8,F8,7C,
```

```
3C,3C,1C,1C,1C,1C
2120 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0
2130 DATA 3,3,3,3,3,3,3,3,1,1,1,1,0,3,4,
4
2140 DATA 80,0,F,3F,78,F0,E0,0,0,1,7,9C,
B0,80,88,86
2150 DATA 0,0,80,E0,78,E,0,0,0,F8,E,1F,3
F,3F,3E,1C
2160 DATA 0,0,0,0,0,0,0,0,0,0,0,0,80,0,0
,0
2170 DATA 0,0,0,0,7,1C,0,0,0,1,F,18,0,0,
4,3
2180 DATA 0,0,0,FC,FF,F,3,1,0,F8,E,F,1F,
1F,1F,E
2190 DATA C,C,C,C,C,8C,EC,EC,C,C,C,8C,CE
,99,19,11
2200 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0
2210 DATA 4,6,2,2,2,2,2,1,0,0,1,1,1,3,2,
2
2220 DATA 81,C0,40,40,40,40,40,E0,B0,98,
8F,82,82,83,81,81
2230 DATA F0,0,0,0,0,0,1,1,1,1,80,0,0,0,
0,80
2240 DATA 0,0,0,0,40,C0,80,0,0,A0,31,1F,
0,39,7F,FF
2250 DATA 0,0,0,0,40,60,38,8,8,D8,80,0,0
,E0,F0,F8
2260 DATA F8,0,0,0,0,0,0,0,0,0,7,0,0,1,1
,3
2270 DATA 33,22,32,12,12,12,12,1C,30,60,
C0,80,80,80,0,0
2280 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0
2290 DATA 6,4,4,C,8,8,F,0,0,0,0,0,0,0,0,
0
2300 DATA 80,80,80,80,80,80,80,0,0,0,0,0
,0,0,0,0
2310 DATA C1,47,61,30,10,18,8,C,4,6,2,3,
1,0,0,0
2320 DATA C0,0,C0,70,7F,3F,1F,F,0,0,0,0,
80,C0,6F,38
2330 DATA 3F,6,3C,F8,F0,E0,E0,E0,C0,0,0,
```

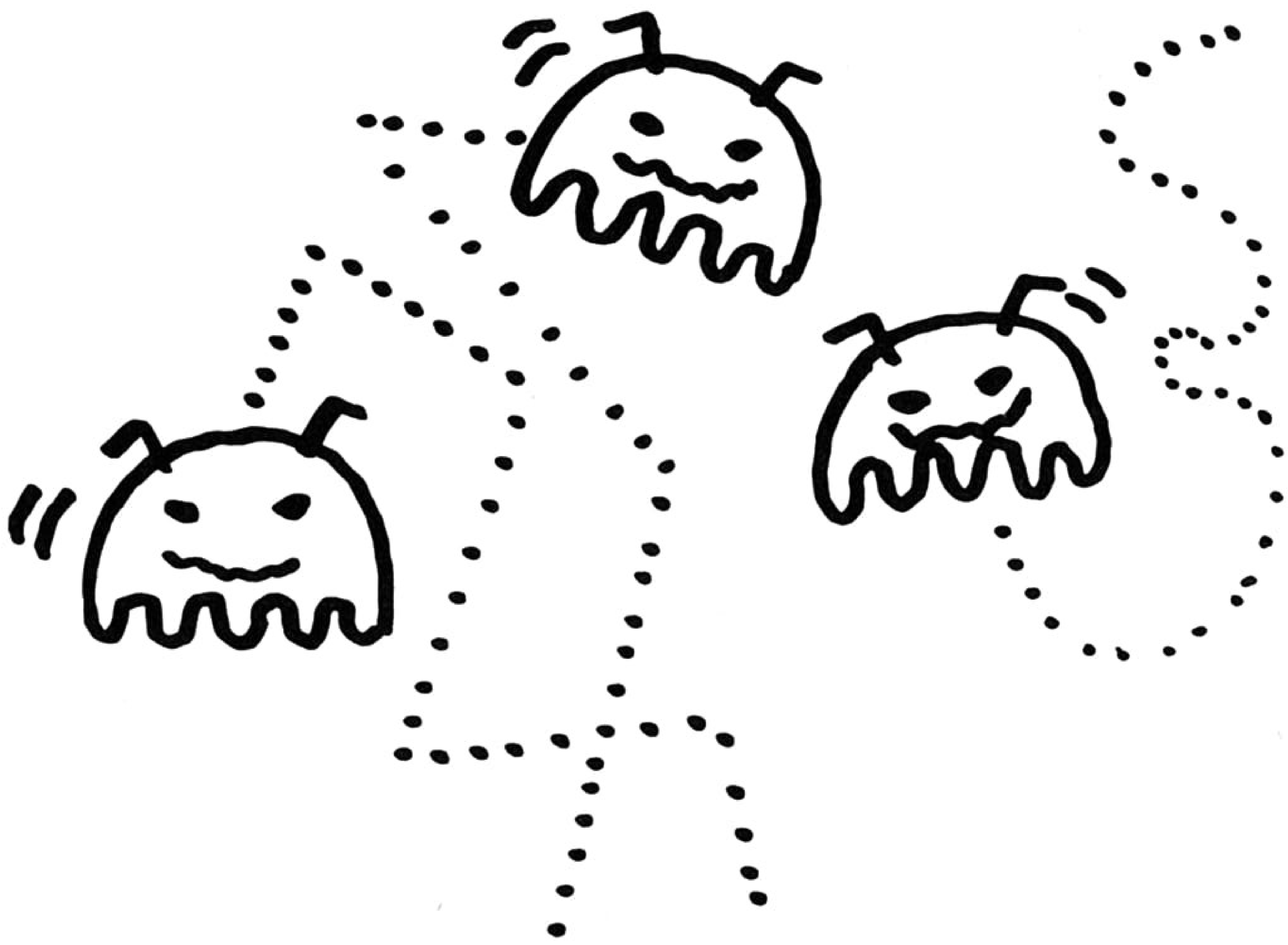
```
1,3,6,C,B8,E0
2340 DATA 6,4,C,18,10,30,60,40,C0,80,80,
0,0,0,0,0
```

このゲームは15パズルといい，テレビゲームとしてはあまり見かけませんが，オモチャ屋のパズルコーナーではよく見られるものです．

ゲームのやり方は単純そのもので，最初に表示されている絵を覚えておき，16個にわかれた破片を動かして，もとの絵にすれば完成です．ただし，破片は15個しかなく，破片のないところへは隣りあった破片をスライドさせて入れることができます．破片を移動させる方法はこれしかありません．移動は，カーソル移動キーを使います．

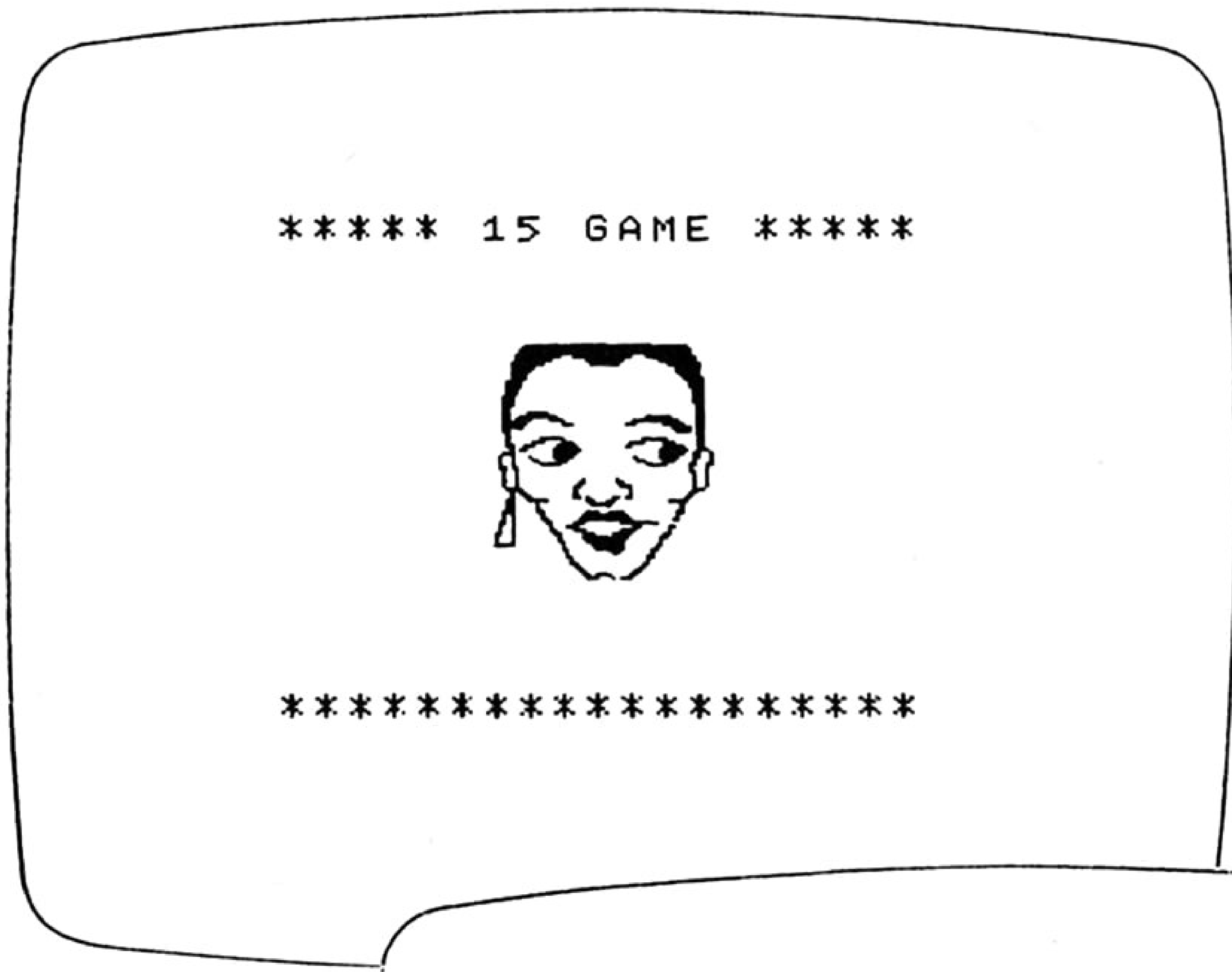
慣れてしまうと簡単なゲームですが，最初のうちはいくらか頭を使うかもしれません．絵が完成したらスペースバーを押してください．絵が本当に完成しているかどうかをチェックして，それまでにかかった時間を表示します．

今まで見たようなことができるのが，スプライト機能ですが，いかがですか？どんなことに使うか，もう考えをめぐらしている人もいると思います．しかし，この章はほんのさわりですから，スプライト機能についてふれていないこともかなりあります．あるゲームキャラクタをなめらかに動かすだけが，スプライトの機能ではないのです．以降の章では，そういったことを含めて，表示の仕方やゲームキャラクタの作り方などについて，例をあげて説明していきます．

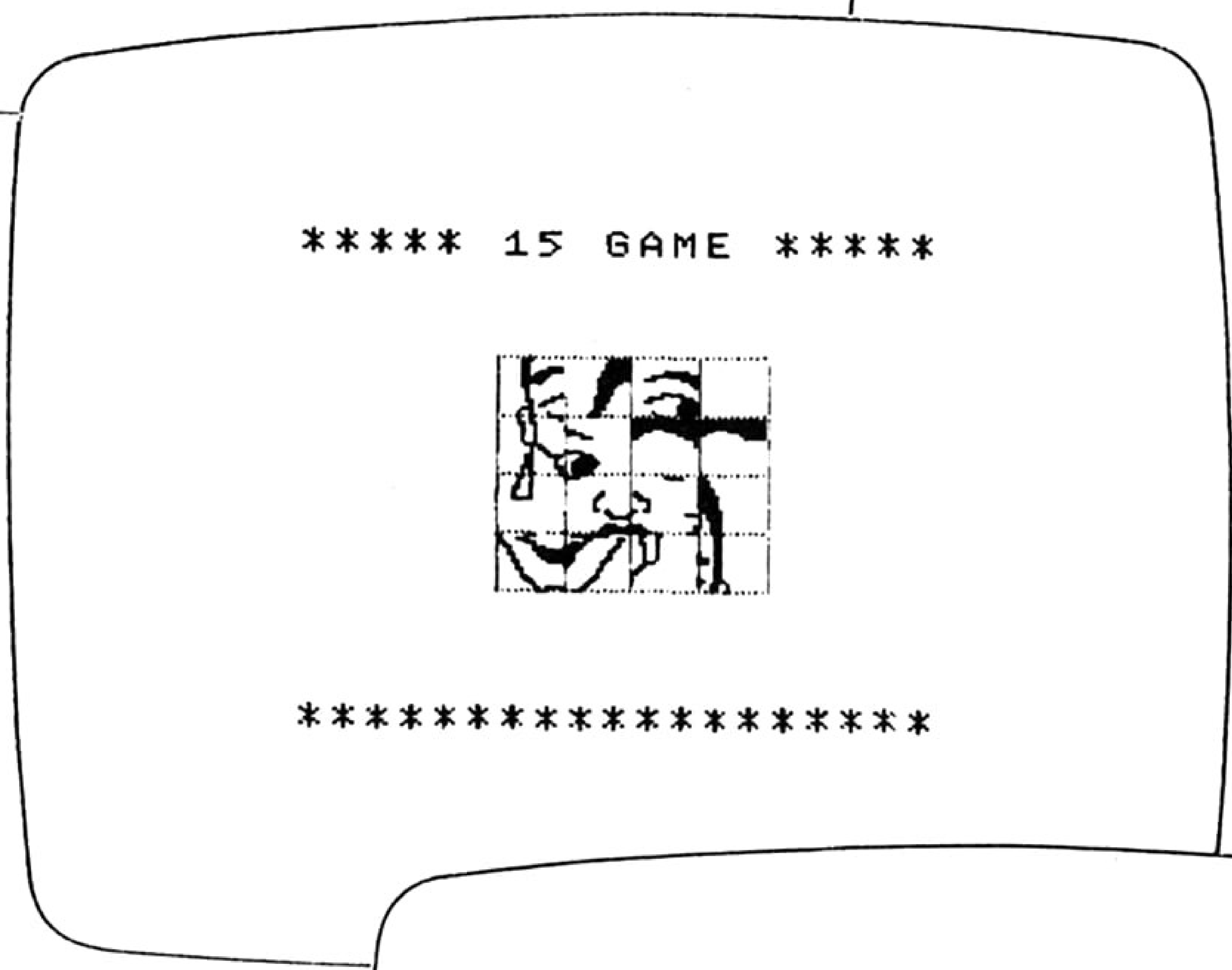


## 15パズル画面写真

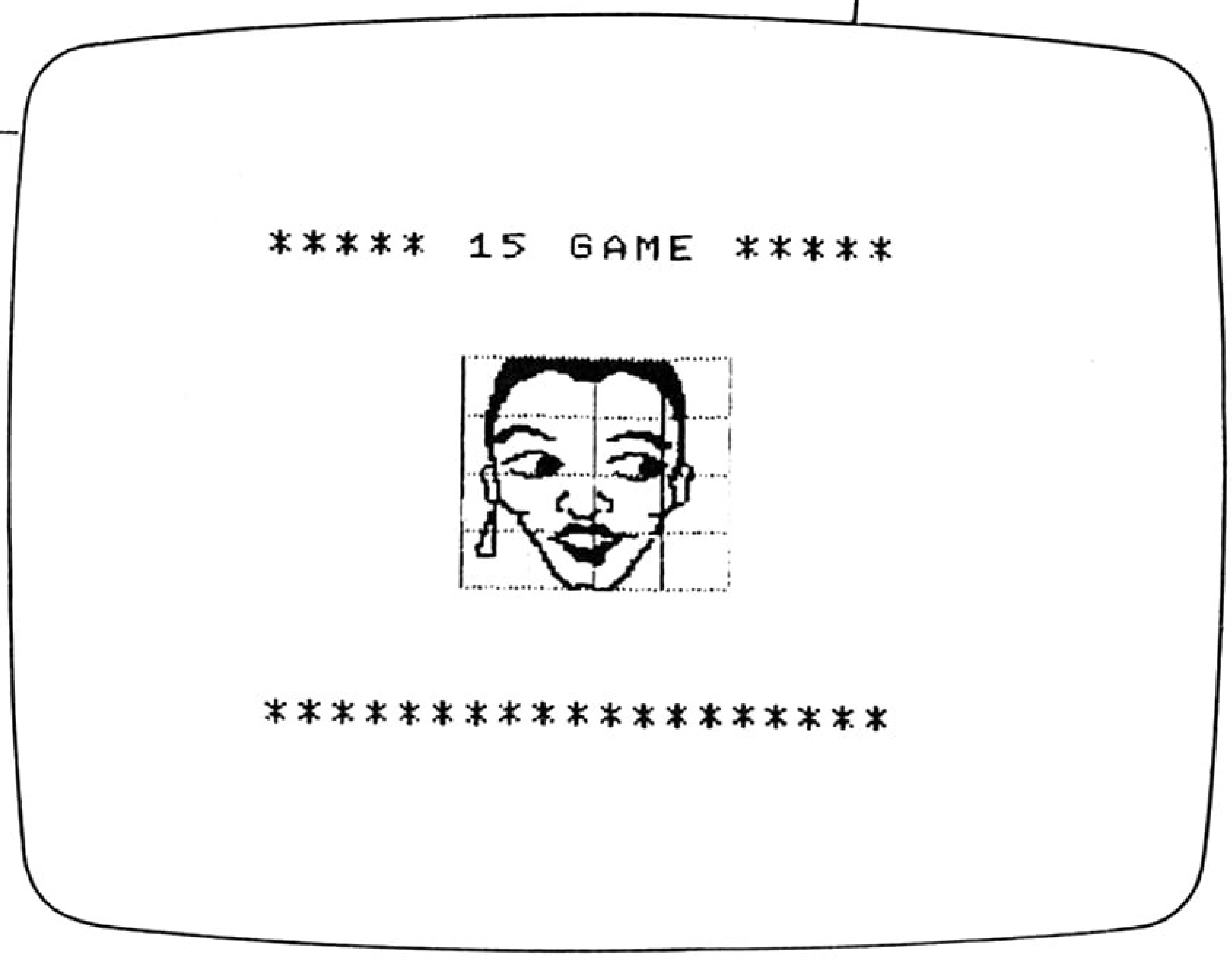
画面１－１




画面１－２




画面１－３



GAME CHARACTER

# 2 スプライトの秘密



スプライト機能を利用してゲームを作るにしても，やみくもに使っているばかりではあまりよい結果を期待できません．スプライトの性質をよく知ってこそ，効果的な使い方ができるのです．この章ではゲームキャラクタ操縦術をマスターする上で，どうしても知っておかなければならない基礎知識を学んでみましょう．

# 1 ゲームキャラクタ=パターン+プレーン

本のタイトルになっている「ゲームキャラクタ」という言葉は，文字通りゲームの中で画面を動きまわる主役たちをさしています．たとえば，昔大流行したインベーダーやパックマンなどがそうです．MSXでは，このようなゲームキャラクタの形を自由に設計して登録しておくことができます．登録した形のことを，MSXの用語で「**スプライト・パターン**」と呼びます．

また，あらかじめ登録しておいたスプライト・パターンを呼び出すには，「**スプライト・プレーン**」というものを使う必要があります．スプライト・プレーンは，キャラクタなどのパターンを画面上に表示させるための，パイプの役目を果すものです．具体的にはプレーンの上に，登録したパターンを載せて，画面上を動き回らせることで，従来のパソコンにはなかった特殊なグラフィックス機能を実現することができるのです．

# 2 パターンの性質

**パターンのサイズは2種類**

スプライト・パターンは2種類のサイズを持っています．8×8ドットと16×16ドットです．基本的にはこの2つのサイズのマス目を適当に塗りつぶして，パターンの形を作っていきます．

**2倍に拡大表示できる**

また，この2つのサイズはそれぞれ縦横2倍に拡大して表示することができます．ですから，見た目の大きさだけを考えると，3つのサイズということになります．しかし32×32ドットでスプライト・パターンを設計することはできません．

**256種までパターンを登録できる**

スプライト・パターンは，最高256種のパターンを登録することができます(16×16ドットでは64種まで)．そのおかげで，かなりおもしろい動きを表現することができます．たとえば，パターンの形が少しずつ変化しながら，画面上を動くアニメーションのような動きも可能になるわけです．

リスト2-1

```
100 COLOR 4,1,1:SCREEN 2,0:CLS
110 GOSUB 260
120 FOR J=5 TO 250 STEP 10
130  FOR I=0 TO 5
140   X=I+J:Y=90
150   PUT SPRITE 0,(X,Y),,I
160   FOR W=0 TO 30:NEXT
170  NEXT
180 NEXT
190 END
200 '
210 DATA 6,"♠♠♠♠♠♠♠♠^@","♠♠♠@♠♠♠^@"
220 DATA "♠♠@ @♠♠^@","♠@ ^P @♠^@"
230 DATA "@ ^P^P^P @^@"
240 DATA " ^P^P^P^P^P ^@"
250 '
260 READ PN
270 FOR J=0 TO PN-1
280  READ D$:I=0:S$=""
290  IF I>=LEN(D$) THEN 340
300   I=I+1:F=0
310   IF MID$(D$,I,1)="^"
      THEN I=I+1:F=64
320   D=(ASC(MID$(D$,I,1))-F)AND 255
330   S$=S$+CHR$(D):GOTO 290
340  SPRITE$(J)=S$
350 NEXT
360 RETURN
```

この例では，8×8ドットのパターンを表示しており，行番号100のSCREEN文にある「0」を「1」に変えると1つの点の大きさを拡大して表示します．こうすることによって，8×8ドットの構成が多少わかりやすくなるかもしれません．

# 3 プレーンの性質

一方，プレーンにもいくつか注意しなければならないことがあります．実際，スプライト機能を使うときには，このプレーンの性質が大きく左右します．

**プレーンは32枚ある**

その1つ目は，プレーンが32枚あることです．それぞれには，0から31までの番号がふってあり，まず不足するということはないでしょう．次のプログラムを実行してみてください．

リスト2-2

```
10 SCREEN 1
20 SPRITE$(0)=STRING$(8,255)
30 PUT SPRITE 32,(12,90),,0
40 SCREEN 0
50 PRINT "PUT SPRITE !!"
60 END
```

画面には，Illegal function call in 30という表示が出たと思います．これは，行番号30にエラーがあることを意味しています．そこで行番号30の「32」という数字を「31」になおしてみましょう．エラーはなくなり，画面には，「PUT SPRITE!!」という表示が現われます．みなさんのなおした箇所は，プレーンの番号を指定するところなので，いま述べたようにプレーンの番号は，31までしかないことがわかります．0番からはじまっていることは，みなさんが確認してみてください．

**1つのプレーンには1つのパターン**

プレーンを扱う上での2つ目の注意は，パターンとプレーンとの関係についてです．一言で説明してしまえば，「1つのプレーンには1つのパターンしか表示できない」ということです．

一見，なんでもないようなこと——ともすると，1つのプレーンにたくさんの

パターンが表示できないことは不便なのではないか，と考えてしまうこの事実は，その反面スプライト機能にとって重要なことなのです．

このおかげで，次のパターンを表示すると同時に前のパターンが消え，瞬時に移動するような，すばやい動きが実現できるのです．

プレーンの枚数を知る方法は，各プレーンに１つずつパターンを表示してみることです．「１つのプレーンには，１つのパターンしか表示できない」のですから，すべてのプレーンに１つずつのパターンを表示すれば，画面に表示されたパターンの数は，プレーンの数にほかなりません．しかし，これまでにお話した２つの性質に加えて，３つ目の性質がここでは問題になってきます．

### プレーンの優先順位

さきほど述べたように，プレーンには０から31までの番号がふってありますが，この番号は単に区分けのためにふってあるのではなく，優先順位の意味でもあるのです．パターンを表示しているプレーンは，それにつけられた番号の若いほうが，手前にくるように画面上に重ねられてパターンを表示しているのです．

したがって，画面上にパターンが１つしか見えなくても，そのうしろにいくつかのパターンがかくれているかもしれないので，画面に見えているパターンの数が，プレーンの数とはいいきれないのです．

では，表示する位置とパターンの色を少しずつ変化させて表示してはどうでしょう．この考えはいいアイデアですが，さらにもう１つのスプライトの性質が関わってきます．

### パターン表示は横４つまで

すべてのプレーンにパターンがあったとしても，かならず全部が表示されるわけではありません．手前のパターンにかくれて見えないのは当然ながら，水平方向――つまり，横に５つ以上のパターンが並ぶと，手前のプレーンに表示されているパターンから数えて，５つ目以降は表示されません．

この原因はハードウェアによるものなので，変更はできません．ただし，パターンを縦に並べる場合には問題ありませんし，それほど多くのパターンを動かしながら表示したところで，MSXの実行スピードを考えると，何のメリットもありません．ですから，この性質も大きな障害となることはないでしょう．かえって，

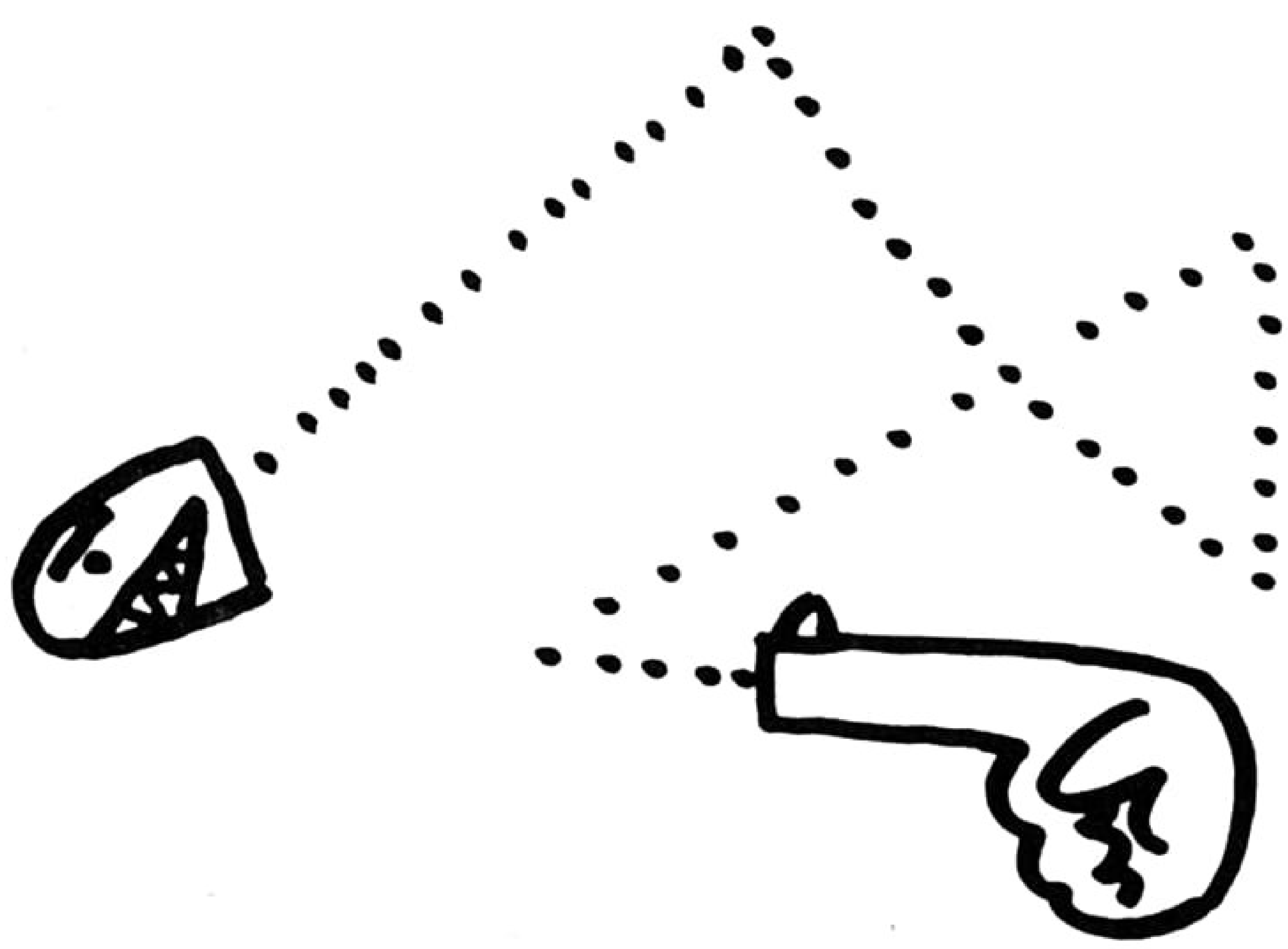
プログラムの実行スピードが落ちるほうが，ゲームなどの場合には大問題になると思います．

## リスト2-3

```
100 COLOR 9,1,1:SCREEN 2,0:CLS
110 SPRITE$(0)=STRING$(8,255)
120 FOR J=0 TO 3
130  FOR I=0 TO 7
140   X=I*15+J*30+30:Y=I*25
150   P=I*4+J:C=(P MOD 14)+2
160   PUT SPRITE P,(X,Y),C,0
170  NEXT
180 NEXT
190 A$=INKEY$:IF A$="" THEN 190
200 END
```

**リスト 2-3** は，いままで述べてきたことに注意して，32個のパターンを表示するプログラムです．

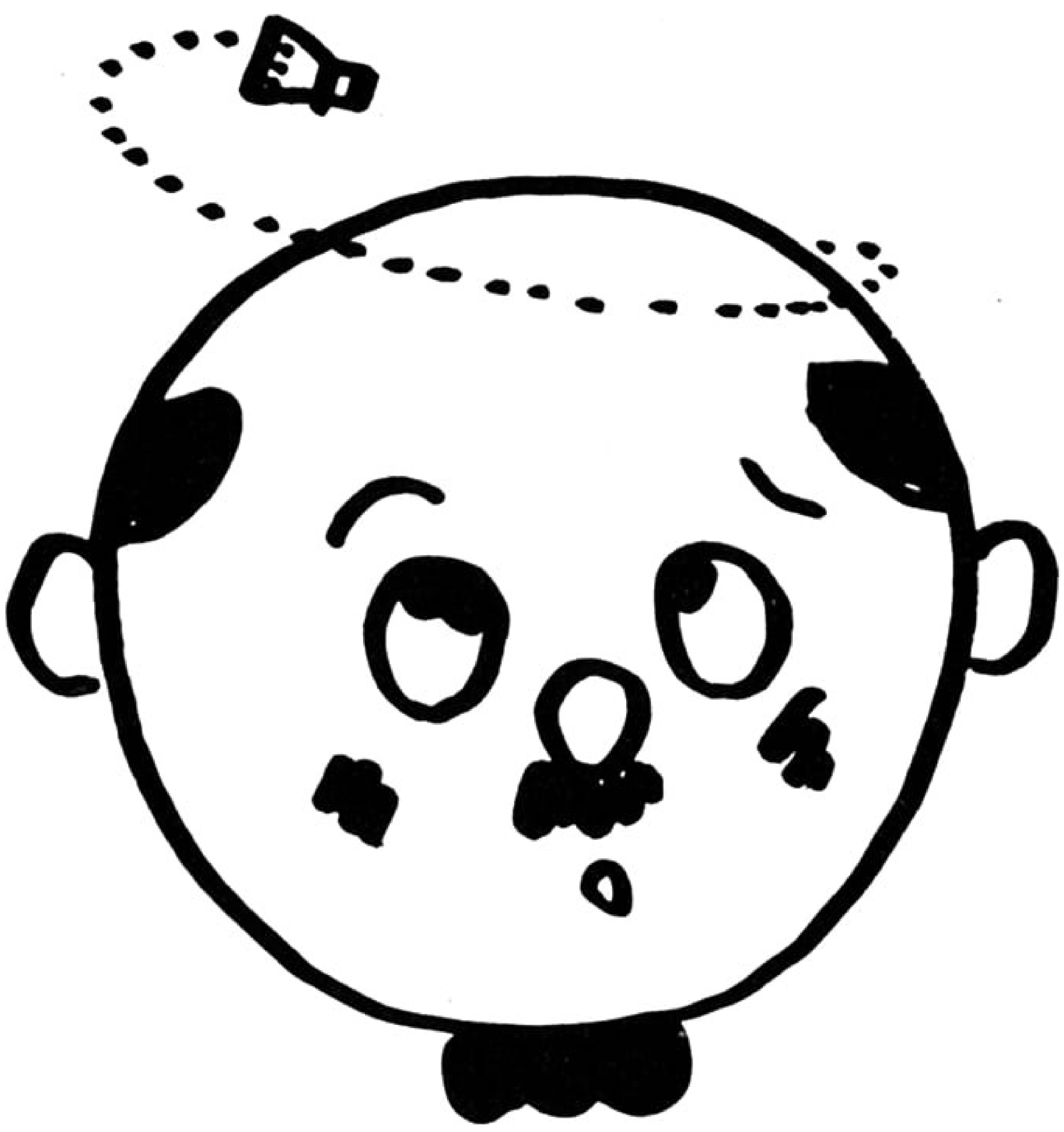
以上のようなスプライト機能は，TMS9918という型のLSIで実現されています．次の章からは，MSX-BASICのコマンドをどのように使って，スプライト機能を操作するのか，ということについて具体的に話を進めていきましょう．



GAME CHARACTER

# 3

# 前準備,そしてキャラクタ作り



いままでは，スプライト機能とはどんなものかを，いくつかのプログラムによって紹介してきました．また，スプライト機能で使用されるパターンやプレーンについてもお話してきましたが，具体的，かつ詳細なことについては，あまりふれませんでした．ですから，「プログラムを組む」という面からは，何も話をしていないことになります．

そこでこの章では，スプライト・パターンを表示する方法や，そのデータの作り方などを１つ１つ具体的に説明しながら，スプライト機能と他のMSX-BASICのコマンドとのつながりを見ていきましょう．

# 1 画面状況を設定するSCREEN文

スプライト・パターンを表示するためには，画面の状況をその目的に応じて設定する必要があります．何を画面に表示するか――これは，スプライト・パターンだけとはかぎりません――，さらにどのようにスプライト・パターンを表示するか，など……画面の状況設定はさまざまです．そこで，MSX-BASICは，それらの指示のためにあるコマンドを持っています．それが**SCREEN文**です．

SCREEN文を用いるのは，この本の目的にかぎればスプライト・パターンを表示する画面の状況を設定するためです．しかし，SCREEN文は，スプライト機能のために用意されているのではなく，他にいくつかMSXの機能を設定できるようになっています．

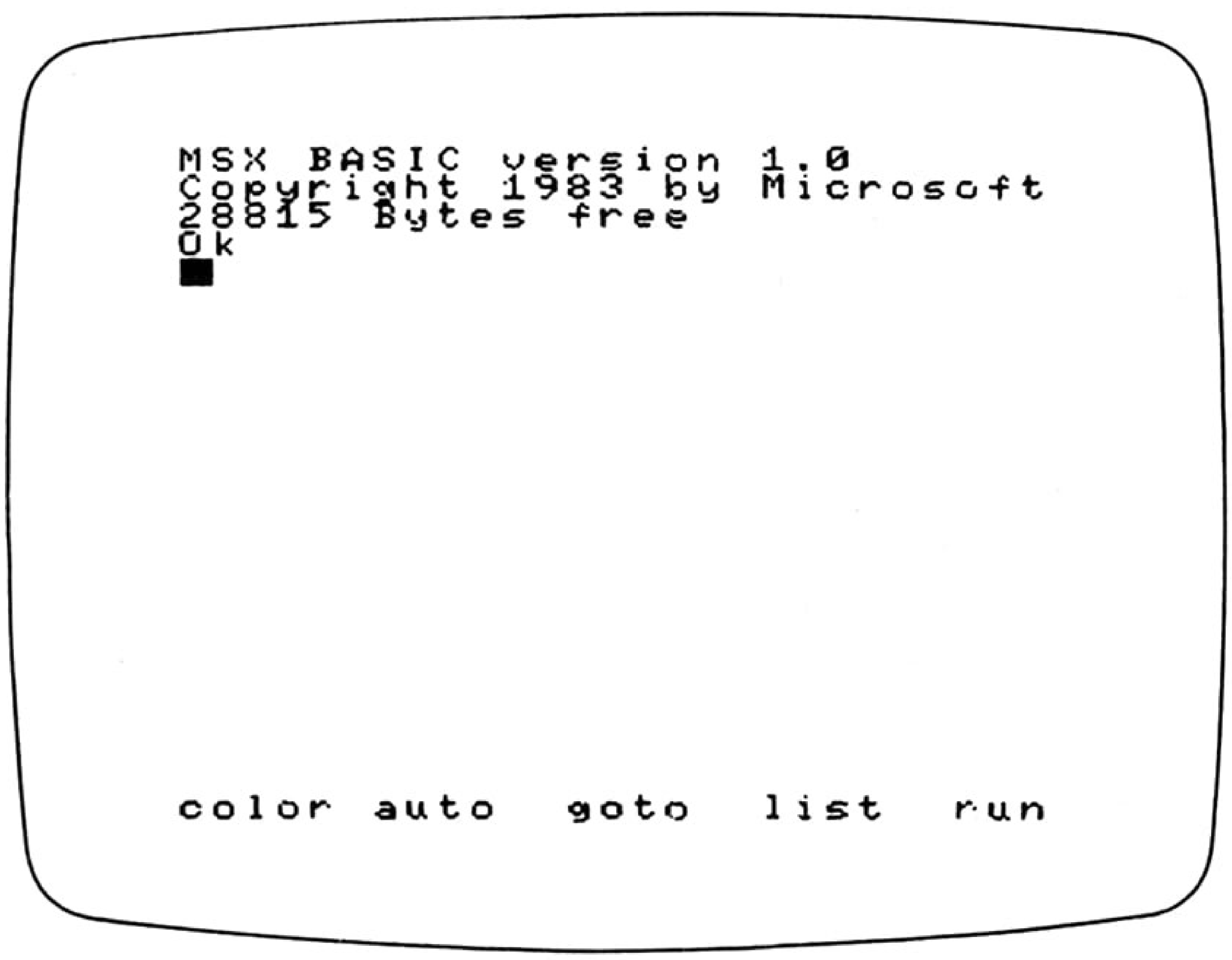


表3－1　SCREEN文による設定内容

| | |
|---|---|
| 画面モード | 0：最大40×24文字のテキストモード |
| | 1：最大32×24文字のテキストモード |
| | 2：256×192ドットの高解像度グラフィックモード |
| | 3：64×48ブロックのマルチカラーモード |
| スプライト・サイズ | 0：8×8ドットのスプライト・パターン |
| | 1：8×8ドットのスプライト・パターン(縦横2倍表示) |
| | 2：16×16ドットのスプライト・パターン |
| | 3：16×16ドットのスプライト・パターン(縦横2倍表示) |
| キークリックスイッチ | 0：音を出さない |
| | 1：音を出す |
| カセットボーレート | 0：1200ボー |
| | 1：2400ボー |
| プリンタオプション | 0：MSX標準プリンタ |
| | 1：MSX標準プリンタ以外 |

さて，電源を入れたばかりの最初の画面は，次のようになっています．

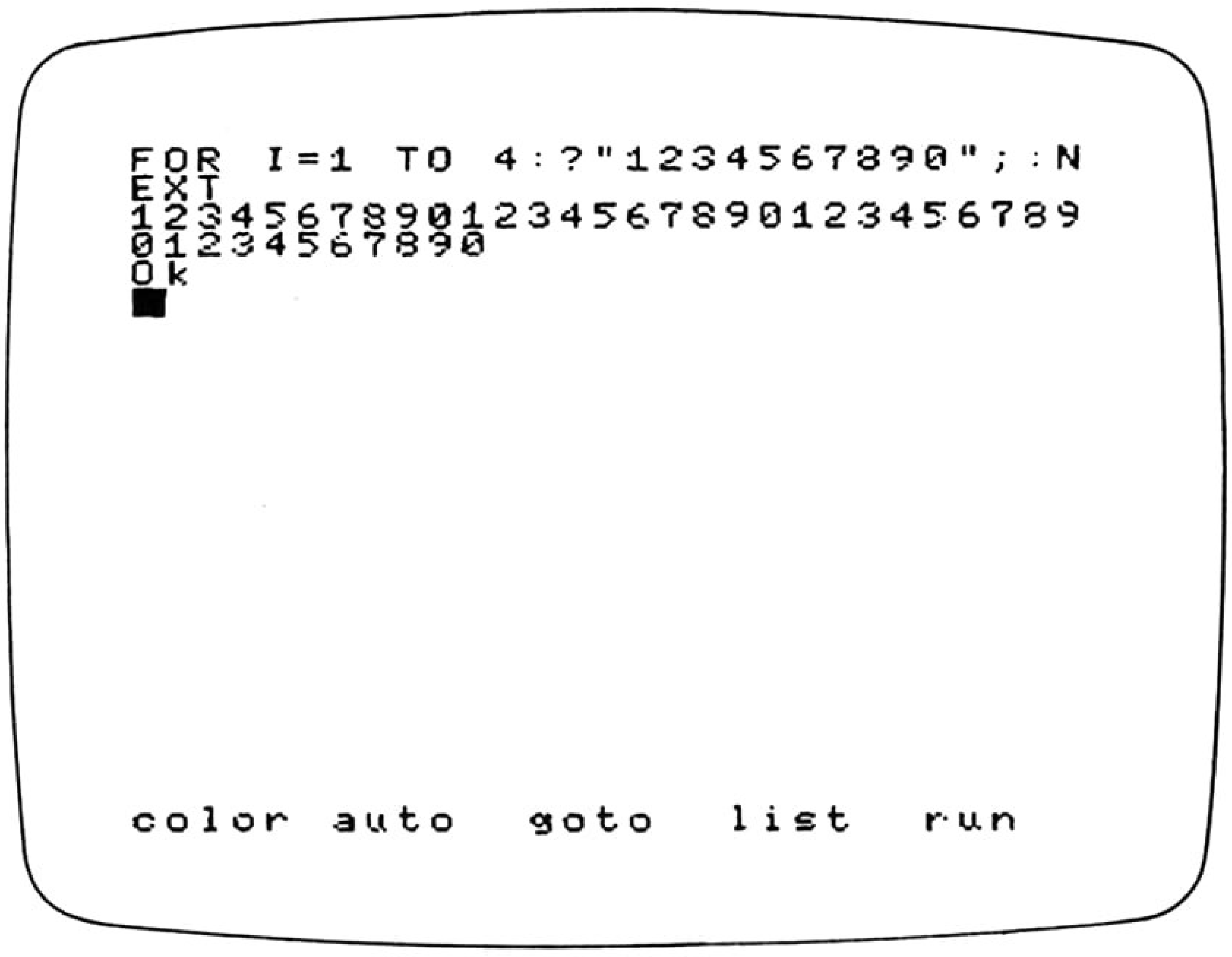
画面3－1




そこで，次のプログラムを入力してみてください．

FOR　I=1　TO　4：？"1234567890"；：NEXT

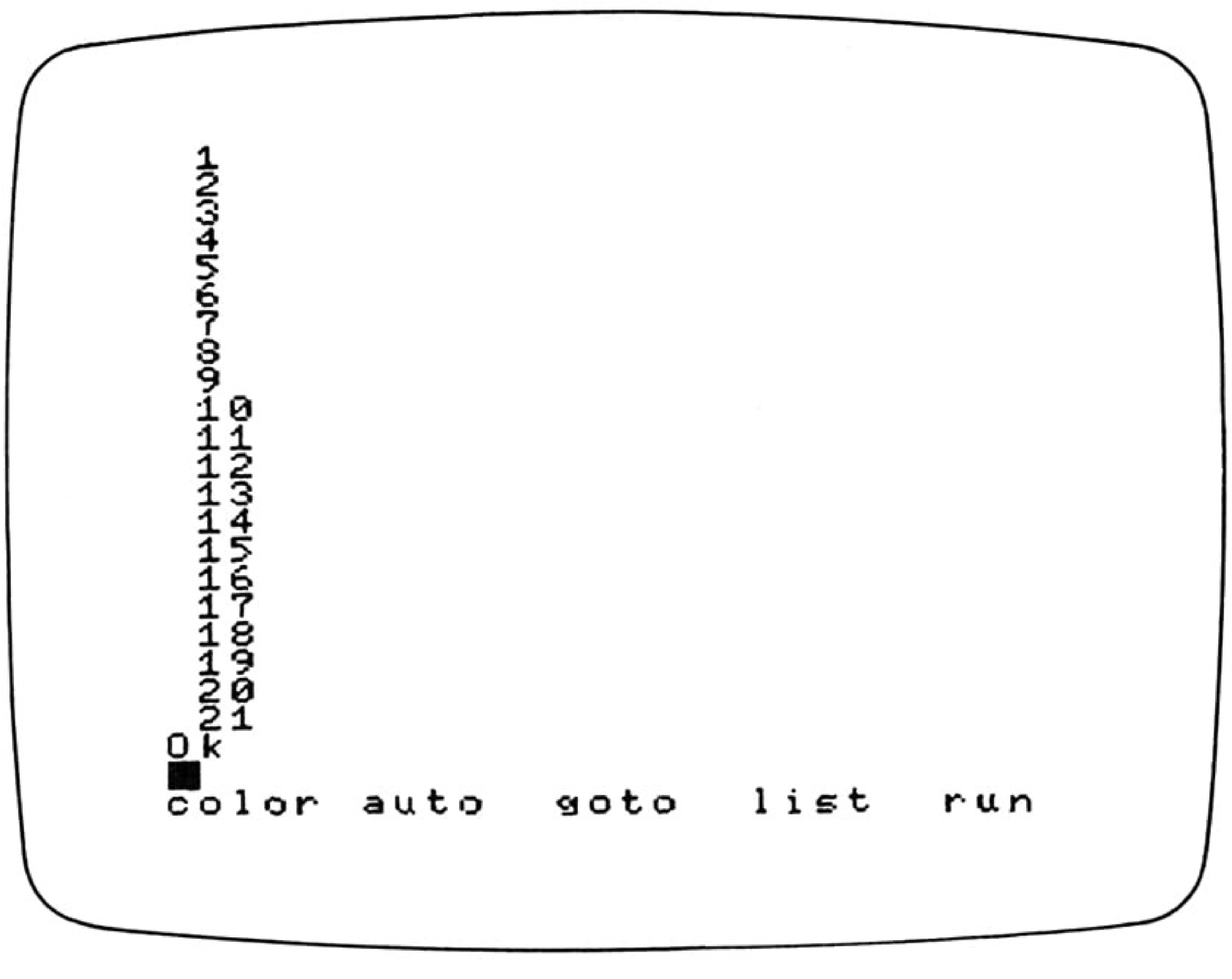
画面 3－2




実行すると，画面 3－2のように表示されるはずです．これを見れば，MSXの最初の画面は横に29文字表示できることがわかります．では，次のプログラムを入力してみてください．

FOR　I=1　TO　21：？I：NEXT

画面 3－3



画面に24行表示できることがわかります．画面上では，21までしか数字が出ませんが，〝Ok〞の文字とカーソル，そしてファンクションキーの内容表示(最下行)を合わせると24行になります．しかし，このままでは，一番下の行は，使用しないのも同じです．24行目を使いたいときは，次のコマンドで最下行を消しましょう．

KEY OFF

ファンクションキー一覧をふたたび表示したいときは，「KEY ON」とタイプします．

さて，さきほどからキーボードを打ってプログラムを入力していますが，タイプを打つたびに音が鳴っていることに気づかれたでしょうか？　この音はキークリック音といいますが，画面に気をとられて聞きそびれた人は，もう一度適当にタイプして確かめてください．それでも聞こえないという人は，テレビのボリュームを少し上げて再挑戦です．

じつはこのキークリック音も，画面の最下行と同じように，音を出すか出さないか，指定できます．この指定をするのがSCREEN文です．キーボードから，

SCREEN ,, 0

と入力してください．これでどのキーを押しても，クリック音はしなくなりました．もとに戻すには，

SCREEN ,, 1

とするだけです．

SCREEN文はさまざまな画面状況を設定できますが，何を設定するのかを「,」で区切られた位置とその数字で表わします．この数字を引数といいます．いまの命令では，SCREEN文と数字の間にカンマが2つはいっています．これは，数字が3番目の引数であることを示すためです．

つまり，SCREEN文の3番目の引数は，キークリック音の有無を決めているのです．では，1番目と2番目は何を指定するのでしょうか？

## 2 SCREEN 0ではスプライトは使えない

1番目の引数で設定するものは，「画面モード」と呼ばれるものです．画面モー

ドとは，画面をどのような状態で使用するかを指定するものです．MSX-BASICの画面は，テキストとグラフィックを表示することができます．テキストとは，たとえばBASICのプログラムや，PRINT文を用いて表示する文字のことです．一方グラフィックは，線・円・四角などの任意の図形のことです．テキスト・モードのPRINT文に対し，グラフィックの画面を操作するためには，LINE文やCIRCLE文などを用います．

電源を入れたばかりの状態では，キーボードから打ち込んだ文字がそのまま画面に出るので，「テキストを表示する」状態であることがわかります．このテキスト・モードにするには，次のように入力します．

SCREEN 1

このように，引数が1番目のときは，カンマは必要ありません．このモードは，最大で横32文字，縦24文字の表示が可能なテキスト・モードです．しかし，画面には横29文字しか表示されません．「WIDTH32」を実行すると32文字を表示できますが，文字が画面の端からはみ出すことがあるので，もとのままのほうがよいでしょう．

横方向にもっと文字数を増やしたい人は，

SCREEN 0

とすれば，最大40×24文字の表示が可能になります．では，**リスト3-1**を実行してみてください．

## リスト3-1

```
10 COLOR 15,4,7:SCREEN 0:CLS
20 FOR I=0 TO 7
30  READ D$
40  S$=S$+CHR$(VAL("&B"+D$))
50 NEXT
60 SPRITE$(0)=S$
70 PUT SPRITE 0,(120,90)
80 END
90 '
100 DATA 10000001
```

```
110 DATA 11111111
120 DATA 10000001
130 DATA 01000010
140 DATA 00100100
150 DATA 00011000
160 DATA 00011000
170 DATA 01111110
```

実行すると，画面には「Illegal function call in 60」と表示されるはずです．これは行番号60に何かのエラーがあるという表示ですが，ここで実際に問題になっているのは行番号10のSCREEN文です．

SCREEN 0は，40×24文字のテキスト・モードを指定する命令ですが，このモードではスプライト機能が使えないのです．行番号10を次のようになおして実行してみましょう．

```
10 COLOR 15, 4, 7 : SCREEN 1 : CLS
```

画面の中央に，何かパターンが現われたと思います．したがって，同じテキスト・モードの画面でも，画面モードが1（SCREENの1番目の引数が1）の場合にはスプライト・パターンを表示できることがわかりました．

一方，テキスト・モードではないグラフィック・モードの状態では，スプライト・パターンが表示できるでしょうか？

## 3 グラフィック・モード上の注意

グラフィック・モードは，画面モードを示す引数の値を，2もしくは3にすることによって指定することができます（つまりSCREEN 2またはSCREEN 3のとき）．では，行番号10を次のように換えて実行してみましょう．

```
10 COLOR 15, 4, 7 : SCREEN 2 : CLS
```

画面には，スプライト・パターンが表示されているようには見えません．しかし，SCREEN 0のときのように，何かエラーが起ったわけでもなさそうです．では，どこにパターンは表示されたのでしょう．

よく画面を観察すると，プログラムの実行が終了した時点では，〝Ok〟という「文

字」が表示されており，さらにキーボードから入力した文字はそのまま画面に表示される状態にあることがわかります．要するに，MSX-BASICはプログラムを実行し終えると，テキスト・モードに戻ることになっているのです．**リスト3-1**の行番号10を変更したプログラムは，実行中はグラフィック・モードですが，実行中に表示されたスプライト・パターンは，テキスト・モードに戻ると消えてしまうのです．パターンをそのまま表示させておくためには，常にプログラムが実行中でなければなりません．そこで，

```
80  GOTO  80
```

として，ふたたび実行すれば画面にパターンが現われます．グラフィック・モードであり，またプログラム実行中なので，当然〝Ok〟という文字は表示されません．プログラムは CTRL ＋ STOP を押して停止させてください．

以上のことはSCREEN 3 についても同じですから，

```
10  COLOR  15, 4, 7 : SCREEN  3 : CLS
```

として，試してみてください．

画面モードを示す値が2と3の場合，グラフィック・モードになることがわかりましたが，この2つにはどのような違いがあるのでしょうか．いままで，見てきた例でわかるとおり，スプライト・パターンに関しては何も影響はないようです．その違いは，グラフィック命令のときに表われます．

**リスト3-2**

```
10 COLOR 15,4,7:SCREEN 2:CLS
20 LINE(0,0)-(255,191)
30 LINE(255,0)-(0,191)
40 CIRCLE(120,90),60
50 LINE(10,10)-(245,181),,B
60 GOTO 60
```

このプログラムは基本的なグラフィック命令を組み合わせた例ですが，行番号10を見るとわかるように，SCREEN 2の条件下で実行されています．SCREEN 2は，グラフィック・モードの中でも，高解像度グラフィック・モードと呼ばれる

ものて，画面上の画素数(ドットの数)は，256×192ドットになっています．座標指定の方法は，**図 3－1**のとおりです．**リスト 3－2**の実行結果からわかるように，割合細かい点で絵を描くことができます．しかし，**図 3－2**(a)に示したような1ブロックの中では，2色しか色が出せません．そこで，1つの点にもっといろいろな色を使ってみたい場合に用意されているモードがSCREEN 3です．

**図 3－1**

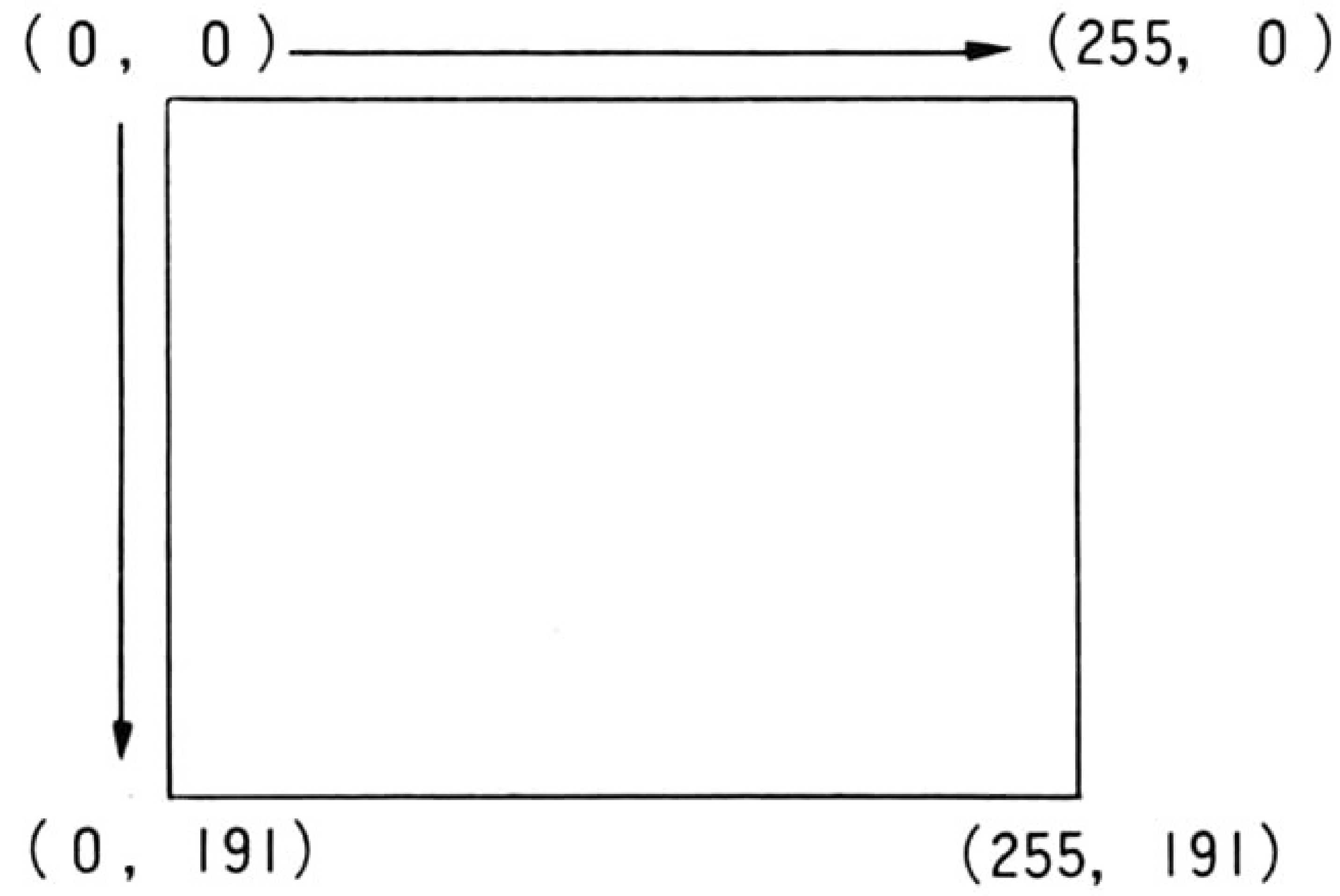
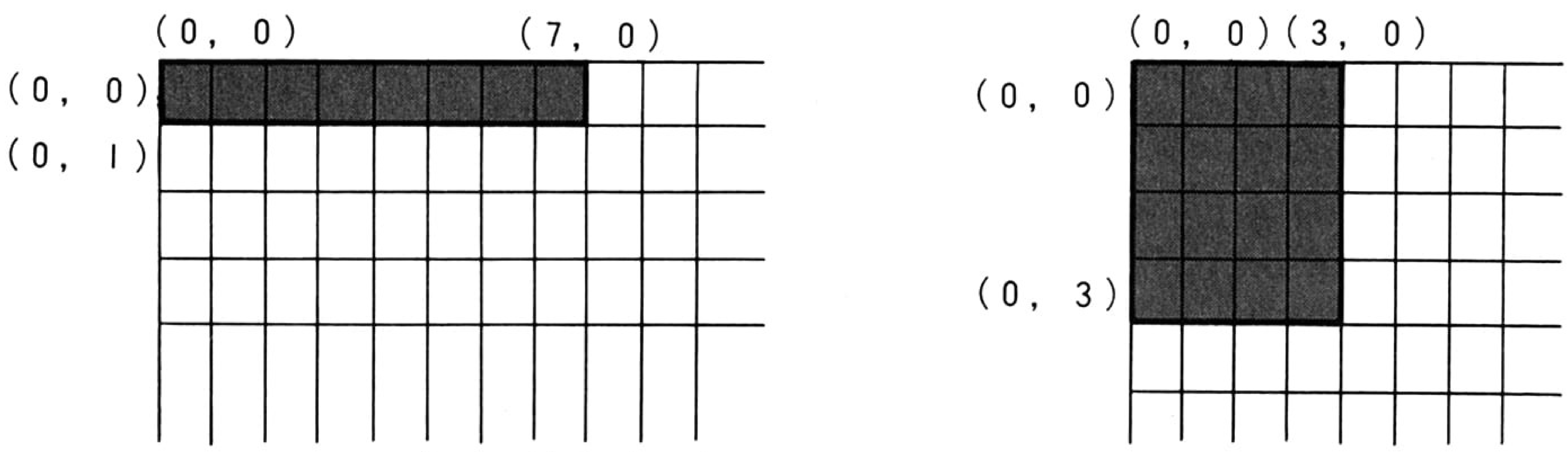



**図 3－2**




(a)太線内が1ブロック　　(b)

座標の指定の方法は，**図 3－1**に示したように，高解像度グラフィック・モードと同じです．しかし，点の大きさが**図 3－2**(b)に示した大きさなので，実際に図形を描くと64×48ドットの解像度しかありません．

どんな絵になるのか，**リスト 3－2**のプログラムの行番号10を次のように換えて実行してみてください．

```
10  COLOR15, 4, 7 : SCREEN 3 : CLS
```

画面モードについていろいろお話してきましたが，スプライト・パターンは，グラフィック・モードでなくても(SCREEN 1はテキスト・モード)表示できることから，スプライトがまったくのグラフィック命令というわけではないことがわかったと思います．実際，スプライト機能を制御しているLSI(TMS9918)は，テキスト画面の文字表示などの機能も果たしているので，テキスト画面上でスプライト・パターンが表示できても，まったく不思議ではないのです．また，純粋なグラフィック命令でないというのは，グラフィック命令を使って描く絵がある画面と，スプライト・プレーンが別のものであることからも説明できると思います．したがって，グラフィック命令で描いた絵を背景として，その手前でスプライト・パターンを動かすこともできるのです．

# 4 パターンのサイズとSCREEN文

さて，スプライトには直接関係のない引数についてふれてきましたが，次にSCREEN文中で直接スプライトを操作する２番目の引数についてお話しましょう．この引数は，スプライト・パターンのサイズやその表示状態(普通の表示と拡大表示)の設定をします．

ここでふたたび，**リスト３−１**のプログラムを見てください．SCREEN 0では，エラーが出ますから，行番号10は，

```
10 COLOR 15, 4, 7 : SCREEN 1 : CLS
```

となおします．このプログラムは，行番号100以降を見てもわかるとおり，8×8ドットのスプライト・パターンを表示するプログラムです．ではさらに，行番号10を，

```
10 COLOR 15, 4, 7 : SCREEN 1, 0 : CLS
```

として実行するとパターンはどうなるでしょう．実行すればわかるとおり，パターンの表示はまったく変化しません．したがって，電源を入れてすぐの状態では，２番目の引数は０になっていることがわかります．この状態は，「8×8ドットのスプライト・パターンの通常表示」を意味します．

次に，8×8ドットのパターンの拡大表示をしてみましょう．

```
10 COLOR 15, 4, 7 : SCREEN 1, 1 : CLS
```

16×16ドットのパターンの通常表示，拡大表示をするときは，２番目の引数をそれぞれ，「2」，「3」にすればよいわけです．

これまでの話では，スプライト・パターンを表示するために，画面をどのモードに設定すればよいかを，SCREEN文の使い方を通して説明しました．そこで今度は，スプライト・パターンのデータの作り方や，さまざまな画面のモードでの，具体的な表示の仕方について話を進めていきましょう．

# 5 キャラクタを作る・登録する

いままでのいくつかのプログラムを見て，もしくはMSXのマニュアルを見て，スプライト・パターンのデータの作り方がもうわかっている人がいるかもしれませんが，まず，８×８ドットのスプライト・パターンのデータの作り方について説明します．

たとえば，**リスト３－３**のプログラムを見てください．

リスト3-3

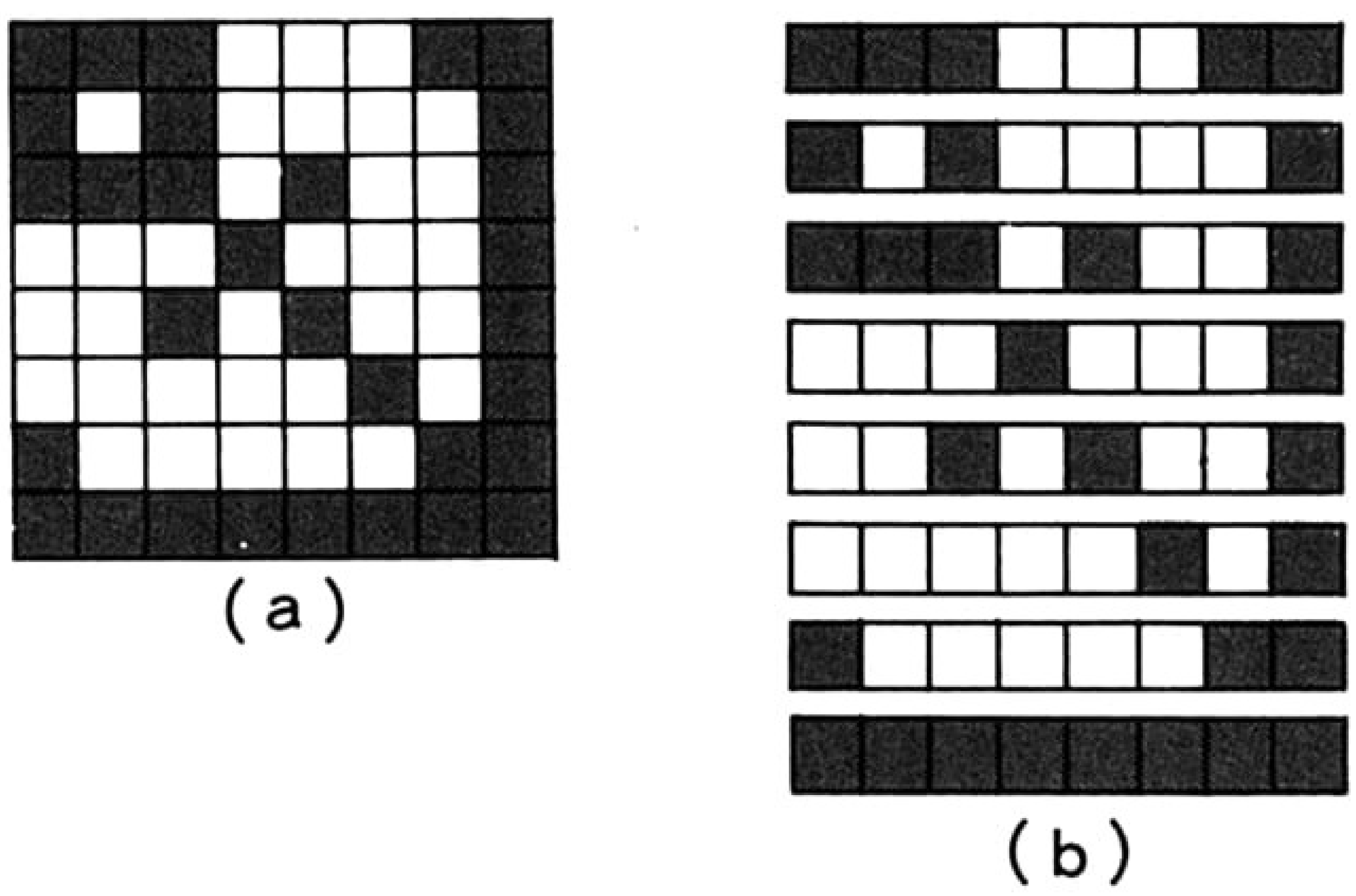
```
100 COLOR 15,4,9
110 SCREEN 1,0:CLS
120 FOR I=1 TO 8
130  READ D$
140  S$=S$+CHR$(VAL("&B"+D$))
150 NEXT
160 SPRITE$(0)=S$
170 PUT SPRITE 0,(120,90),15,0
180 END
190 '
1000 DATA 11100011
1010 DATA 10100001
1020 DATA 11101001
1030 DATA 00010001
1040 DATA 00101001
1050 DATA 00000101
```

```
1060 DATA 10000011
1070 DATA 11111111
```

プログラムの構造は**リスト3−1**とまったく同じです.このプログラムの行番号100から180は，行番号1000以降のデータ(8×8ドット)を表示する機能を持っています.したがって,ここで説明する8×8ドットのスプライト・パターンをMSX-BASICに登録し，表示するための命令を，すべて含んでいることになります.

8×8ドットのスプライト・パターンの構成は，**図3−3**(a)に示したように考えられます.**図3−3**(a)の灰色に塗ってあるところが，表示されるパターンを表わしています.このような形でパターンができ上がったら，次に**図3−3**(b)のように横方向8ドットずつに分解します．そして，それぞれ灰色の部分を1，白い部分を0として数字と対応させると，**図3−3**(c)のように，簡単に2進数の形でデータに変換できたことになります.

**図3−3**




```
1 1 1 0 0 0 1 1
1 0 1 0 0 0 0 1
1 1 1 0 1 0 0 1
0 0 0 1 0 0 0 1
0 0 1 0 1 0 0 1
0 0 0 0 0 1 0 1
1 0 0 0 0 0 1 1
1 1 1 1 1 1 1 1
```

(c)

この2進数のデータは，数値ではなく，文字列です．そして**リスト3−3**の行番号120から150にあるFORループで，この文字列を数値に変換し(VAL)，さらにこれをキャラクタ・コードの文字列になおして(CHR$)，文字変数S$の中に取り込んでいます．この文字列をスプライト・パターンのデータとして，MSXに登録するには，行番号160のようにSPRITE$に代入してやります．SPRITE$は，システム変数と呼ばれるものです．カッコでくくられた添字は，「何番目のスプライト・パターンとして登録するか」ということを意味し，8×8ドットのパターンでは，添字として0から255の値を指定することができます．表示の際には，この添字で

望みのスプライト・パターンの番号を指定することができます．したがって，**リスト3－3**のプログラムでは，０番にスプライト・パターンを登録したことになります．

# 6 パターンの表示

ひとたび登録されれば，そのパターンはPUT SPRITE文を用いて，スプライト・プレーンにPUTする(＝置く→表示する)ことができます．表示するときに，指示する必要があるのは，次のとおりです．

**(1) 表示する位置**
**(2) 表示するスプライト・パターン**
**(3) PUTするスプライト・プレーン**
**(4) 表示する時の色**

(2)に示した「表示するスプライト・パターン」は，SPRITE$のところでお話したように，希望するスプライト・パターンの番号を指定します．(3)も同じように，０から31までのプレーンの番号を指定します．

「表示する位置」は座標を用います．その座標は，**図3－4**に示したような範囲の数値で示されるもので，スプライト・プレーンの一部が画面上に表示されます．

**図3－4**

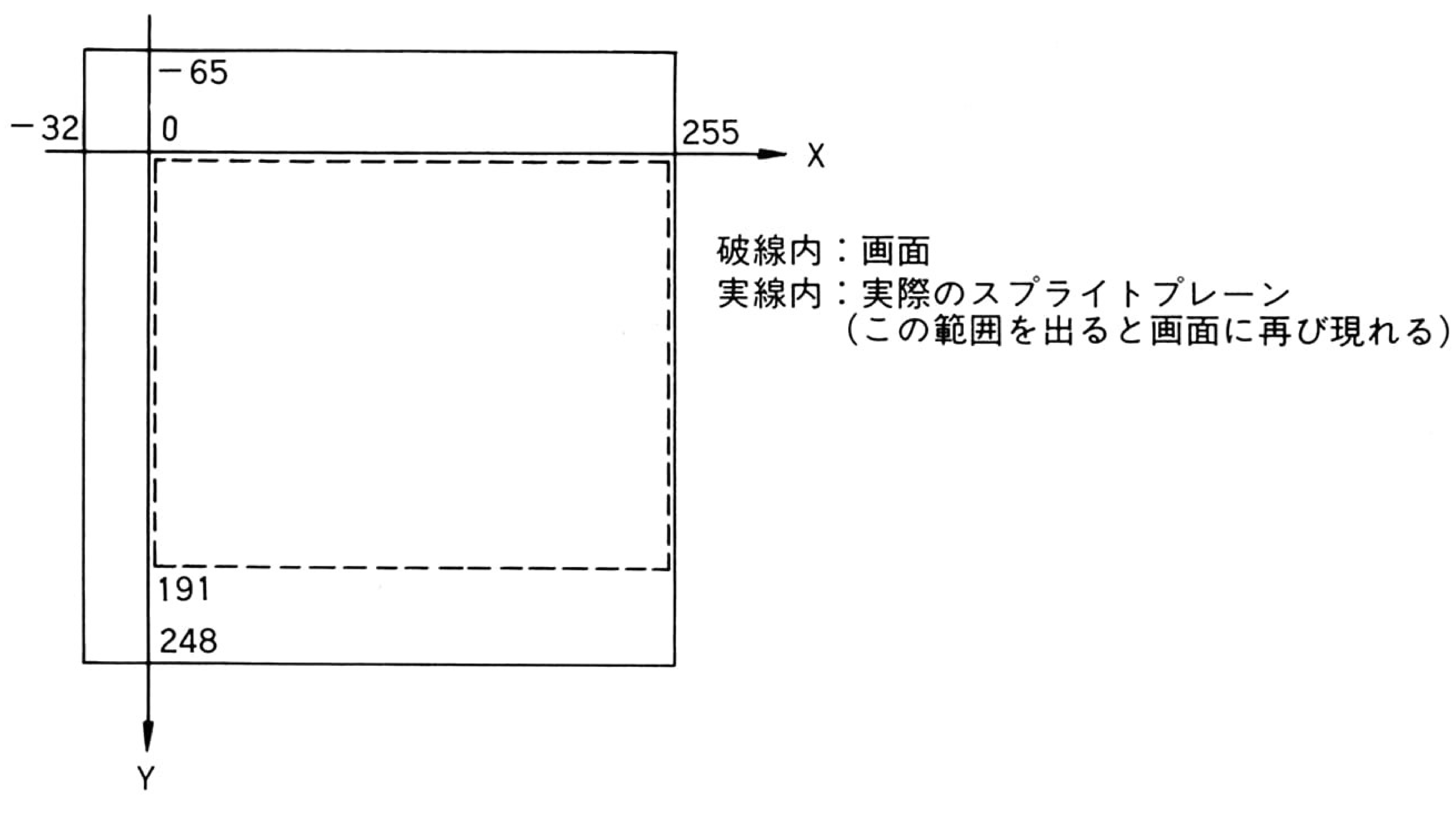


図 3－4 に示したように，一般に，横方向の軸をx軸，縦方向の軸をy軸と呼び，それぞれの座標をx座標，y座標と呼びます．このx，y各座標の値を用いて，表示する位置を指定します．MSXで表示するときには，指定された座標に，スプライト・パターンの左上の点を合わせて表示します．

以上，スプライト・パターンを表示するときに，指定するものをあげてきました．それらを命令文(ステートメント)の形で書くと，**リスト 3－3** の行番号170のようになります．与えている数値は，次のような意味を持っています．

```
170 PUT SPRITE 0,(120,90),15,0
```

- 0：スプライト・プレーンの番号
- 120：表示する位置のx座標の値
- 90：表示する位置のy座標の値
- 15：表示するときの色(カラー・コード)
- 0：スプライト・パターンの番号

# 7 表示法の2つの応用例

PUT SPRITE文で指定する数値の中で，スプライト・プレーンの番号以外は，すべて省略することができます．省略したときに，カラー・コードの場合はCOLOR文で設定した前景色になります．残りのものは，スプライト・プレーンが現在持っている値が使用されます．したがって，表示位置とスプライト・パターンの番号は，最後に参照されたものがプレーン別に管理されていることがわかります．ただし，座標は，省略する際にカッコごと省略する必要があります．

たとえば，

```
PUT SPRITE 0,,9
```

のようにです．

また，パターンの最後の表示位置をおぼえているので，そこからの相対的な位置を示すために，次のような書き方もできます．

```
PUT SPRITE 0, STEP(2,2), 15, 0
```

この書き方ができるおかげで，ある場合には無駄な計算を省略することができます．たとえば，**リスト3－4**の(a)と(b)とを見比べてください．これは，飛行機が左のほう（画面では，下のほう）に飛んでいくところを上から見たときのサンプルプログラムですが，(a)の行番号145の計算は，(b)のほうではPUT SPRITE文でSTEPを用いているので，不要になっています．

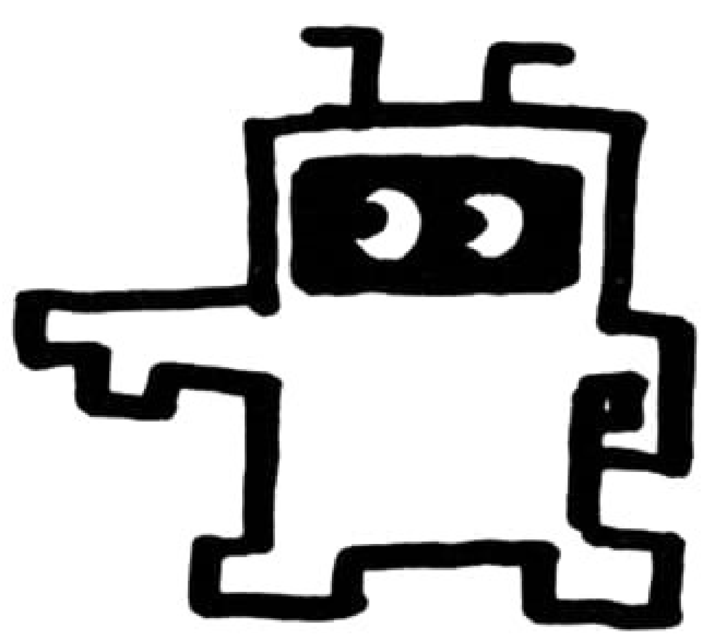
なお，行番号100のSCREEN文は，8×8ドットの2倍表示を指示しているのがわかります．

### リスト3-4(a)

```
100 COLOR 15,4,7:SCREEN 1,1:CLS
110 SPRITE$(0)="001°°100"
120 X=255:Y=60
130 PUT SPRITE 0,(X,Y)
140 FOR I=0 TO 128
145  X=X-2:Y=Y+1
150  PUT SPRITE 0,(X,Y)
160 NEXT
170 END
```

### リスト3-4(b)

```
100 COLOR 15,4,7:SCREEN 1,1:CLS
110 SPRITE$(0)="001°°100"
120 X=255:Y=60
130 PUT SPRITE 0,(X,Y)
140 FOR I=0 TO 128
150  PUT SPRITE 0,STEP(-2,1)
160 NEXT
170 END
```



PUT SPRITEは，スプライト・プレーンとスプライト・パターンを独立に扱うことができるので，異なったスプライト・プレーンに，同じパターンをたくさん表示することもできます．

## リスト3-5

```
100 COLOR 15,1,1:SCREEN 2,1:CLS
110 GOSUB 280
120 FOR I=0 TO 100
130  X=256*RND(-TIME)
140  M=20*RND(-TIME)
150  FOR J=0 TO M:NEXT
160  Y=192*RND(RND(-TIME))
170  PSET(X,Y)
180 NEXT
190 '
200 FOR I=0 TO 24 STEP 2
210  FOR J=0 TO 1
220   X=256*RND(-TIME)
230   PUT SPRITE I+J,(X,I*16),4,0
240  NEXT
250 NEXT
260 GOTO 260
270 '
280 FOR I=1 TO 8
290  READ D$:D=VAL("&B"+D$)
300  S$=S$+CHR$(D)
310 NEXT
320 SPRITE$(0)=S$
330 RETURN
340 '
350 DATA 00000000
360 DATA 00000000
370 DATA 00011000
380 DATA 00111100
390 DATA 01111110
400 DATA 00100100
410 DATA 11111111
420 DATA 01100110
```

**リスト 3-5**のようなプログラムを作るとき，気をつけなければならないのは，

パターンが水平方向に5個以上並ばないようにすることです．この点に気を配っておかないと，期待とは異なった結果になり，表示されないパターンができてしまいます．またここでは，UFOのバックの星を描くためにグラフィック命令を用いて点を打っているので，SCREEN文では高解像度グラフィック・モードを指定しています．

## 8 キャラクタデータもう1つの作り方

ところでさきほどの**リスト3－4**では，SPRITE$に代入している値(？)を見ると，スプライト・パターンを登録するときに，データを文字列の形で与えていることがわかります．

これについて少し考えてみましょう．まず，これを2進数の形で表現すると——つまり，**リスト3－3**のプログラムのDATA文のように示すとどのようになるのか，試してみましょう．

### リスト3-6

```
100 T$="001°°100"
110 FOR I=1 TO 8
120 T=ASC(MID$(T$,I,1))
130 PRINT RIGHT$("0000000"+BIN$(T),8)
140 NEXT
150 END
```

**リスト3－6**の行番号120からわかるように，作られたパターンの横8ドット分を1文字のキャラクタ・コードに変換しているのです．したがって，これまでは，**図3－3**(c)に示してある2進数をキャラクタ・コードと見なして，それを対応する文字に変換するプログラムが必要でしたが，**リスト3－4**ではその部分がないために，DATA文に何行も費やすということはありません．8文字で構成された文字列で，スプライト・パターンのデータができてしまえば，それにこしたことはありません．

しかし，中にはキーボードから入力して，画面に表示されないキャラクタ・コードもあるのです．コントロール・キーなどは，その典型的な例です．そこで，

コントロール・キーを「^」という文字をつけて表現することにしましょう．たとえば，&H03なら，「^C」のようにです．これは単に，〝C〟のキャラクタ・コード(&H43)から&H40を引くだけの作業ですから，その変換用のプログラムさえ作れば，DATA文を0と1で何行も書かないですむようになります．しかし，パターンの形を変更するときには，かえって面倒になります．

## リスト3-7

```
100 READ PN
110 FOR J=0 TO PN-1
120  READ D$:I=0:S$=""
130 .IF I>=LEN(D$) THEN 180
140   I=I+1:F=0
150   IF MID$(D$,I,1)="^"
      THEN I=I+1:F=64
160   D=(ASC(MID$(D$,I,1))-F)AND 255
170   S$=S$+CHR$(D):GOTO 130
180  SPRITE$(J)=S$
190 NEXT
200 RETURN
```

たとえば，**リスト3-4**で用いた飛行機のパターンを，逆向きにしたパターンを表示するために，2進数でデータを作ってみると，**図3-5**(a)のようになります．これを16進数になおすと同じく(b)のようになります．このデータの中の&H0Cなどは，キーボードから入力しても，画面には文字として出力されないコントロール・コードです．&H0Cが入力されるのは，CTRL+Lが押されたときですから，**リスト3-7**用のデータは，**図3-5**(c)のようになります．では，**リスト3-7**に加えて，**リスト3-8**のプログラムを入力してみてください．

**図3-5**

| (a) | (b) | (c) |
|---|---|---|
| 0 0 0 0 1 1 0 0 | 0 C | ^L |
| 0 0 0 0 1 1 0 0 | 0 C | ^L |
| 1 0 0 0 1 1 0 0 | 8 C | や |
| 1 1 1 1 1 0 1 1 | F B | ろ |
| 1 1 1 1 1 0 1 1 | F B | ろ |
| 1 0 0 0 1 1 0 0 | 8 C | や |
| 0 0 0 0 1 1 0 0 | 0 C | ^L |
| 0 0 0 0 1 1 0 0 | 0 C | ^L |

リスト3-8

```
10 COLOR 15,4,7
20 SCREEN 1,1:CLS
30 GOSUB 100
40 PUT SPRITE 0,(120,90)
50 END
60 DATA 1,"^L^Lゃろろゃ^L^L"
```

実行すれば逆向きの飛行機が表示できるはずです．このようなデータの圧縮は，パターンの種類が少ないときや8×8ドットの場合には，あまり必要ではないかもしれません．

しかし，16×16ドットのパターンなどには，このような表記法も意外に重宝するかもしれません．また，**リスト3－7**および**3－8**を見ておわかりでしょうが，DATA文は最初にパターンの数，その次から〝〟でくくったデータが並ぶようになっています．なお，コントロール・コード以外で，キーボードから入力できないものや，表記が特別なものは，**表3－2**にあげておきます．

**表3－2**

| (コード) | (表記) |
|---|---|
| &H22 | ^b |
| &H5E | ^せ |
| &H7F | ^ソ |
| &H90 | ^ミ |
| &HFE | ^＞ |
| &HFF | ^？ |

これからあとのプログラム中で，パターンの種類が多いものや16×16ドットのパターンなどのときは，DATA文の部分が大きくなるので，2進数でなく16進数を用いることが多くなりますが，このような表現を利用することもあります．

以上のような説明で，8×8ドットのパターンについては，よくわかってもらえたと思います．これさえわかってしまえば，16×16ドットのパターンの場合を理解するのはたやすいことでしょう．そこで，次は16×16ドットのパターンの作

り方などを話していきましょう．

# 9 16×16ドットパターン

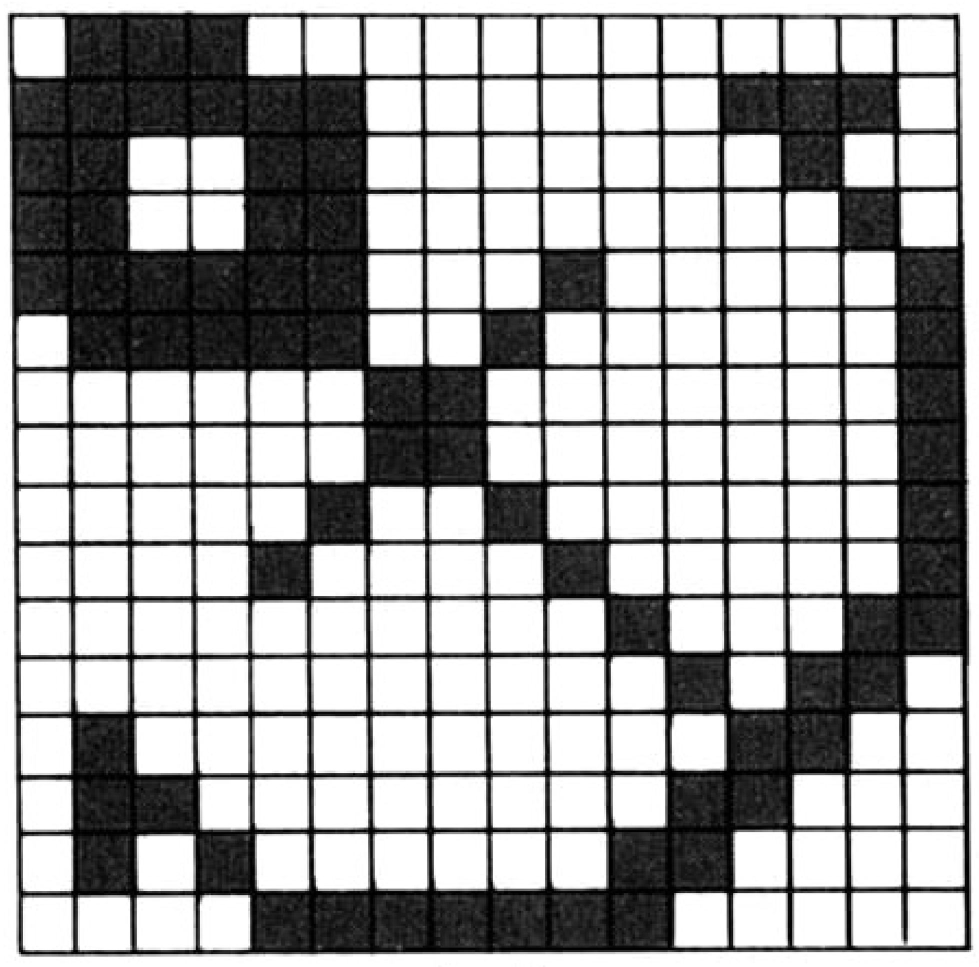
16×16ドットのスプライト・パターンも，一応考え方は８×８ドットと同じです．異なる点といえば，パターンの表現力が多少豊かになったことでしょう．しかし，その構成となると，８×８ドットのものとはまったく違っています．

リスト3-9

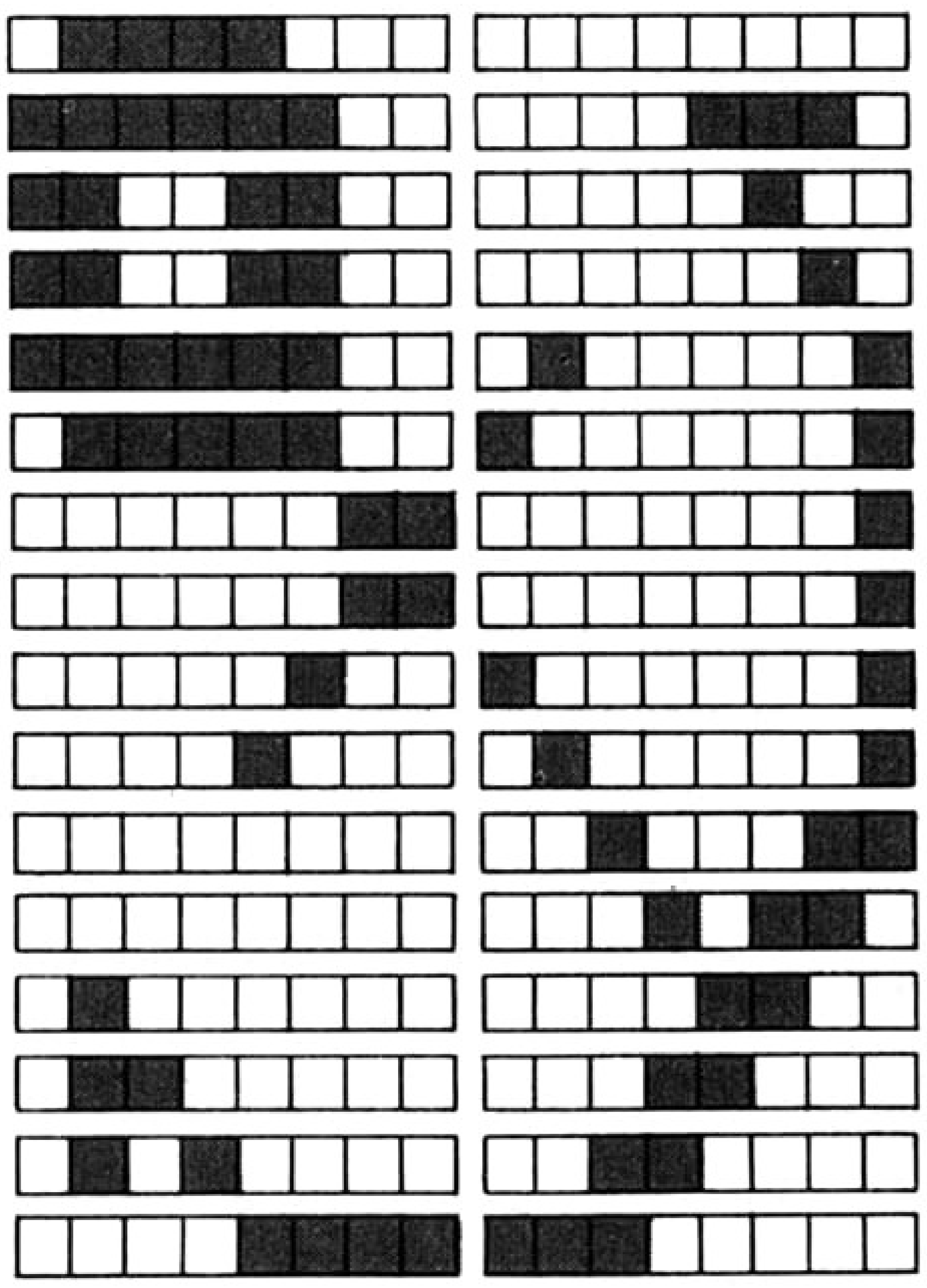
```
100 COLOR 15,4,7
110 SCREEN 1,2:CLS
120 FOR I=1 TO 16
130  READ D$
140  D1=VAL("&B"+LEFT$(D$,8))
150  D2=VAL("&B"+RIGHT$(D$,8))
160  S1$=S1$+CHR$(D1)
170  S2$=S2$+CHR$(D2)
180 NEXT
190 SPRITE$(0)=S1$+S2$
200 PUT SPRITE 0,(120,90),15,0
210 END
220 '
1000 DATA 0111100000000000
1010 DATA 1111110000001110
1020 DATA 1100110000000100
1030 DATA 1100110000000010
1040 DATA 1111110001000001
1050 DATA 0111110010000001
1060 DATA 0000001100000001
1070 DATA 0000001100000001
1080 DATA 0000010010000001
1090 DATA 0000100001000001
1100 DATA 0000000000100011
1110 DATA 0000000000010110
1120 DATA 0100000000001100
1130 DATA 0110000000011000
1140 DATA 0101000000110000
1150 DATA 0000111111100000
```

**リスト3－9**は，**リスト3－3**と同じ構造のプログラムで，DATA文の読み取り方が，16×16ドット用に換えてあります．プログラムを見れば，16×16ドット用のパターンのデータを，どうやって文字列に変換すればよいのかわかると思いますが，より明確にするために，スプライト・パターンの構成を**図3－6**に示しておきます．**図3－6**(a)の灰色で塗ってあるところが，パターンとして表示される点を意味しています．８×８ドットと異なるのは，次の段階です．いままでは，横に８ドットでしたから，単に横長のパターンにわけるだけでしたが，今度の場合は，横に16ドットありますから，これを８ドットずつ，２つの部分にわけます．ちょうど，**図3－6**(b)に示したようになるわけです．したがって，２進数のデータは，自然に**図3－6**(c)になることがわかります．

**図3－6**



(a)



(b)

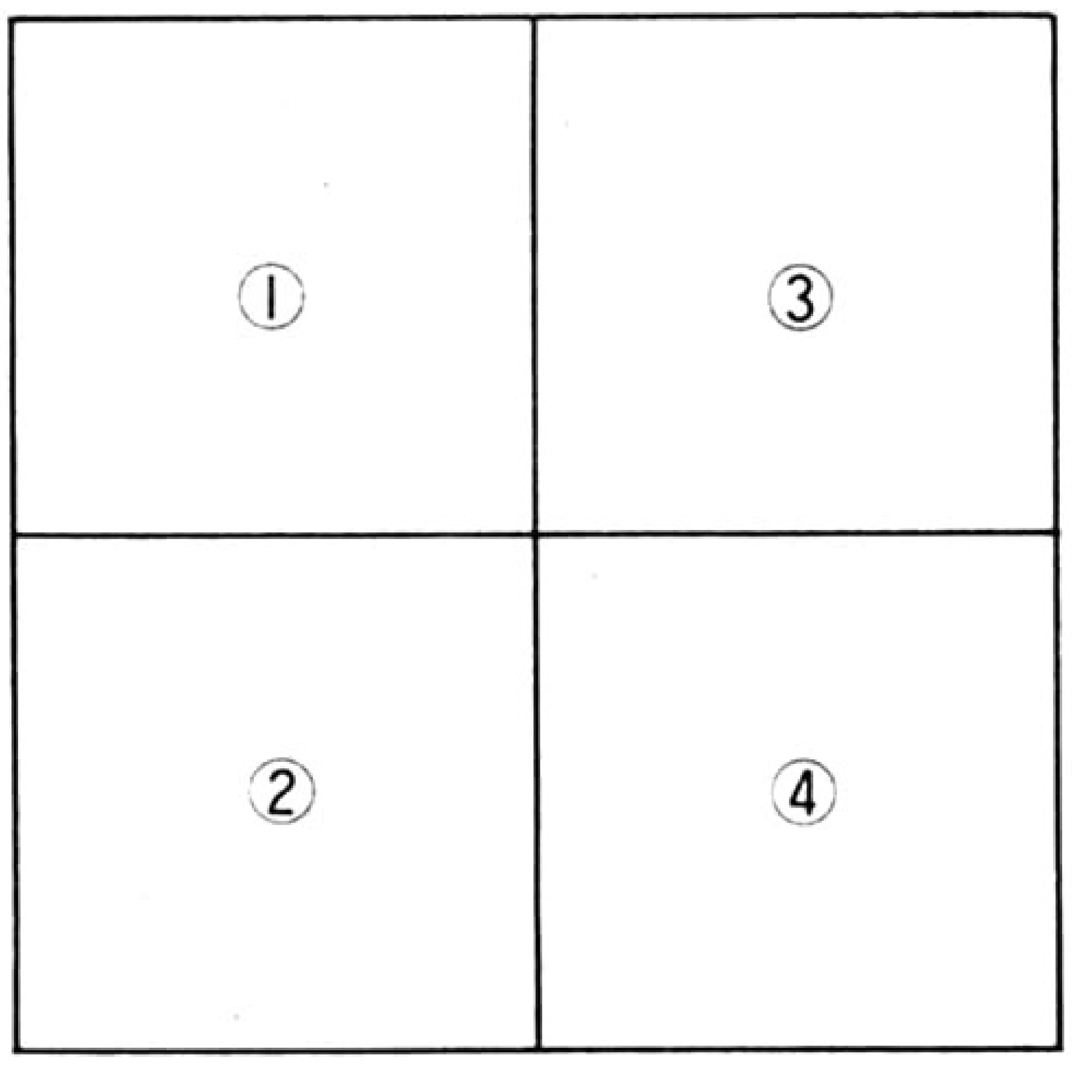
```
0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 0
1 1 0 0 1 1 0 0 0 0 0 0 0 1 0 0
1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 0
1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1
0 1 1 1 1 1 0 0 1 0 0 0 0 0 0 1
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1
0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0
0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0
0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0
0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0
```

(c)



しかし，このデータをただ順番に文字列にしただけでは，まともなパターンにはなりません．16×16ドットの場合には，**図 3－7** に示した順序で文字列にしなければならないのです．

図 3－7




変換された文字列S$は，8×8ドットの場合と同様にSPRITE$に代入することで，MSX-BASICに登録します．使用できる添字は0から63で，合計64個のパターンを同時に登録できます．

登録したパターンを表示するのは，8×8ドットのときに用いた方法とまったく同じです．ただし，SCREEN文は，パターンを登録する前に16×16ドットを表示することができる状態にしておきましょう．

**リスト 3－9** のデータ文を見ればわかるとおり，16×16ドットの場合には非常に膨大なデータで，やっと1つのパターンになっています．1つのプログラム中に何種類もの16×16ドットのパターンが必要なときには，かなりデータがかさばり，下手をするとメモリ不足ということにもなりかねません．そのようなときは，16進数を使ったり，さきほどの**リスト 3－7** および **3－8** のような文字列表現を使うことが，有意義になってくるでしょう．では，その例を1つあげておきましょう．

## リスト3-10

```
100 COLOR 15,1,1:SCREEN 2,2:CLS
110 GOSUB 280
120 FOR I=0 TO 100
```

```
130   X=256*RND(-TIME)
140   M=20*RND(-TIME)
150   FOR J=0 TO M:NEXT
160   Y=192*RND(RND(-TIME))
170   PSET(X,Y)
180  NEXT
190  '
200  FOR I=0 TO 24 STEP 2
210   FOR J=0 TO 1
220    X=256*RND(-TIME)
230    PUT SPRITE I+J,(X,I*16),15,J
240   NEXT
250  NEXT
260  GOTO 260
270  '
280  READ PN
290  FOR J=0 TO PN-1
300   READ D$:I=0:S$=""
310   IF I>=LEN(D$) THEN 360
320    I=I+1:F=0
330    IF MID$(D$,I,1)="^"
       THEN I=I+1:F=64
340    D=(ASC(MID$(D$,I,1))-F)AND 255
350    S$=S$+CHR$(D):GOTO 310
360   SPRITE$(J)=S$
370  NEXT
380  RETURN
390  '
400  DATA 2,"♠♠♠♠ぁ●ゅわ^?ゃ○ぁ♠♠♠♠^A^A^A^Aち。ア
?^?1!ち^A^A^A^A"
410  DATA "0`タ♥ぁ●ぅる^?ゃ○ぁ♥タ`0^L^F^C♥ち。あそ^?
1!ち♥^C^F^L"
```

このようなパターンに，さまざまな動きを与えれば，ゲームとしての基本的な部分ができ上がってしまったことになります．ですから，動きさえ工夫すれば，おもしろいゲームの表示もかなり楽にできてしまうのです．

さて，この章のまとめとして，スロット・マシンのプログラムを組んでみましょう．

## リスト3-11

```
100 COLOR 6,4,7:SCREEN 2,2:CLS
110 DEFINT A-Z:S=1000:N=0
120 DIM D(3),G(3,3,3):GOSUB 740
130 ON STRIG GOSUB 390:STRIG(0) ON
140 '
150 LINE(80,16)-STEP(96,160),3,BF
160 LINE(88,32)-STEP(80,128),9,BF
170 LINE(96,104)-STEP(64,48),5,BF
180 LINE(176,160)-STEP(8,8),14,BF
190 LINE(184,48)-STEP(8,128),15,BF
200 CIRCLE(188,48),10
210 PAINT(188,48)
220 LINE(96,48)-STEP(15,32),1,BF
230 LINE STEP(8,-32)-STEP(16,32),1,BF
240 LINE STEP(8,-32)-STEP(16,32),1,BF
250 LINE(104,120)-STEP(48,24),1,BF
260 COLOR 15
270 OPEN "GRP:" FOR OUTPUT AS #1
280 PRESET(104,128)
290 PRINT #1,S
300 '
310 A=RND(-TIME)*4
320 B=RND(-TIME)*4
330 C=RND(-TIME)*4
340 PUT SPRITE 0,(96,55),D(A),A
350 PUT SPRITE 1,(120,55),D(B),B
360 PUT SPRITE 2,(144,55),D(C),C
370 GOTO 310
380 '
390 IF N<>0 THEN 500
400 A=RND(-TIME)*4
410 PUT SPRITE 0,(96,55),D(A),A
420 N=N+1:RETURN 440
430 '
440 B=RND(-TIME)*4
450 C=RND(-TIME)*4
460 PUT SPRITE 1,(120,55),D(B),B
470 PUT SPRITE 2,(144,55),D(C),C
480 GOTO 440
490 '
500 IF N<>1 THEN 590
```

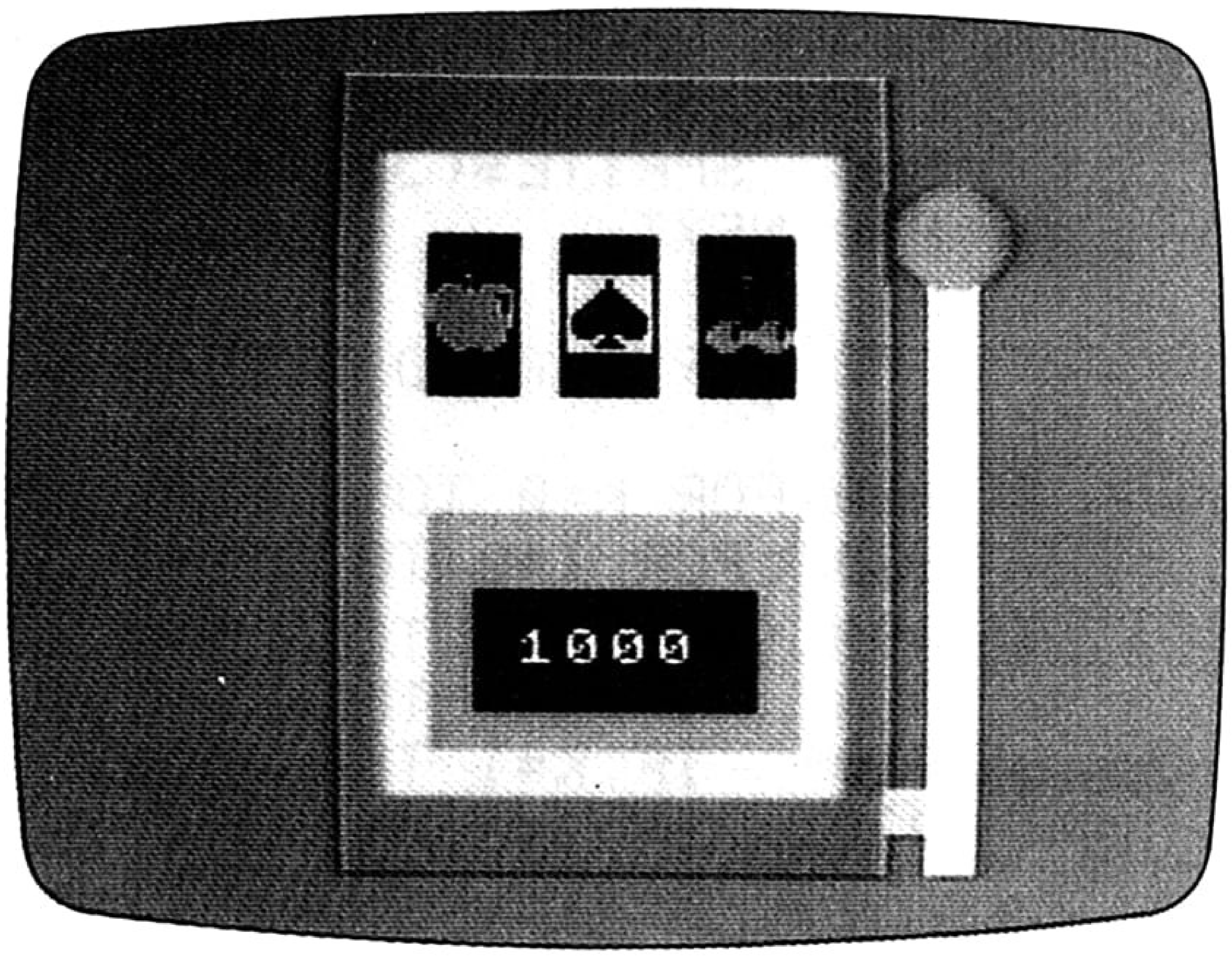
```
510 B=RND(-TIME)*4
520 PUT SPRITE 1,(120,55),D(B),B
530 N=N+1:RETURN 550
540 '
550 C=RND(-TIME)*4
560 PUT SPRITE 2,(144,55),D(C),C
570 GOTO 550
580 '
590 IF N<>2 THEN RETURN 720
600 PUT SPRITE 2,(144,55),D(C),C
610 N=N+1:RETURN 630
620 '
630 S=S+G(A,B,C)
640 LINE(104,120)-STEP(48,24),1,BF
650 PRESET(104,128)
660 PRINT #1,S
670 IF S>0 THEN 670
680 SCREEN 0:CLS
690 PRINT "YOU LOST EVERYTHING !"
700 END
710 '
720 N=0:GOTO 310
730 '
740 READ PN
750 FOR J=0 TO PN-1
760  READ D$:I=0:S$=""
770  IF I>=LEN(D$) THEN 820
780   I=I+1:F=0
790   IF MID$(D$,I,1)="^" THEN
      I=I+1:F=64
800   D=(ASC(MID$(D$,I,1))-F)AND 255
810   S$=S$+CHR$(D):GOTO 770
820  SPRITE$(J)=S$
830 NEXT
840 '
850 FOR I=0 TO 3:READ D(I):NEXT
860 '
870 FOR K=0 TO 3:FOR J=0 TO 3
880  FOR I=0 TO 3:G(I,J,K)=-50
890   IF I=J THEN G(I,J,K)=50*(I+1)
900   IF J=K THEN G(I,J,K)=50*(J+1)
910   IF K=I THEN G(I,J,K)=50*(K+1)
920   IF I=J AND J=K THEN
      G(I,J,K)=200*(I+1)
```

```
930  NEXT
940 NEXT:NEXT
950 RETURN
960 '
970 DATA 4,"^?^>^>わりみたタ♠♠♠♠こわり^?^>^>^>~>
^^^N^F^B^B^B^B=~>^>"
980 DATA "^C^A=^ソ^?^?^?^?^?^?^ソ^ソ^ソ?^_^N
^@^@¦^>めれれれ^>^>^>わわわりp"
990 DATA "^C^A^A^A^A^A^A^B^B>zろろ^?~<♠@ ^
P^P^P^H^H^H<zろろ^?~<"
1000 DATA "^@^A^A^A^A^A^Cぅウヌ^?^ソ6^ソI^ソ^@
♠♠♠♠♠♠タあへて^?^>l^>い^>"
1010 DATA 14,3,6,10
```

持ち点は1000点からです．スペースバーを押すとスタートです．３つのパターンが動きはじめますから，スペースバーを押してください．１回押すごとに左側のパターンから順に停止していきます．リンゴやサクランボなどの組み合わせによって，持ち点が増えたり，減ったりします．

このプログラムはグラフィック・モードで実行されているので，持ち点，得点など文字の出力は，グラフィック上で行なわなければなりません．そのためのテクニックは，詳しく説明しませんが，**リスト３-11**の行番号270～290，630～660のあたりで使用しているので参考にしてみてください．

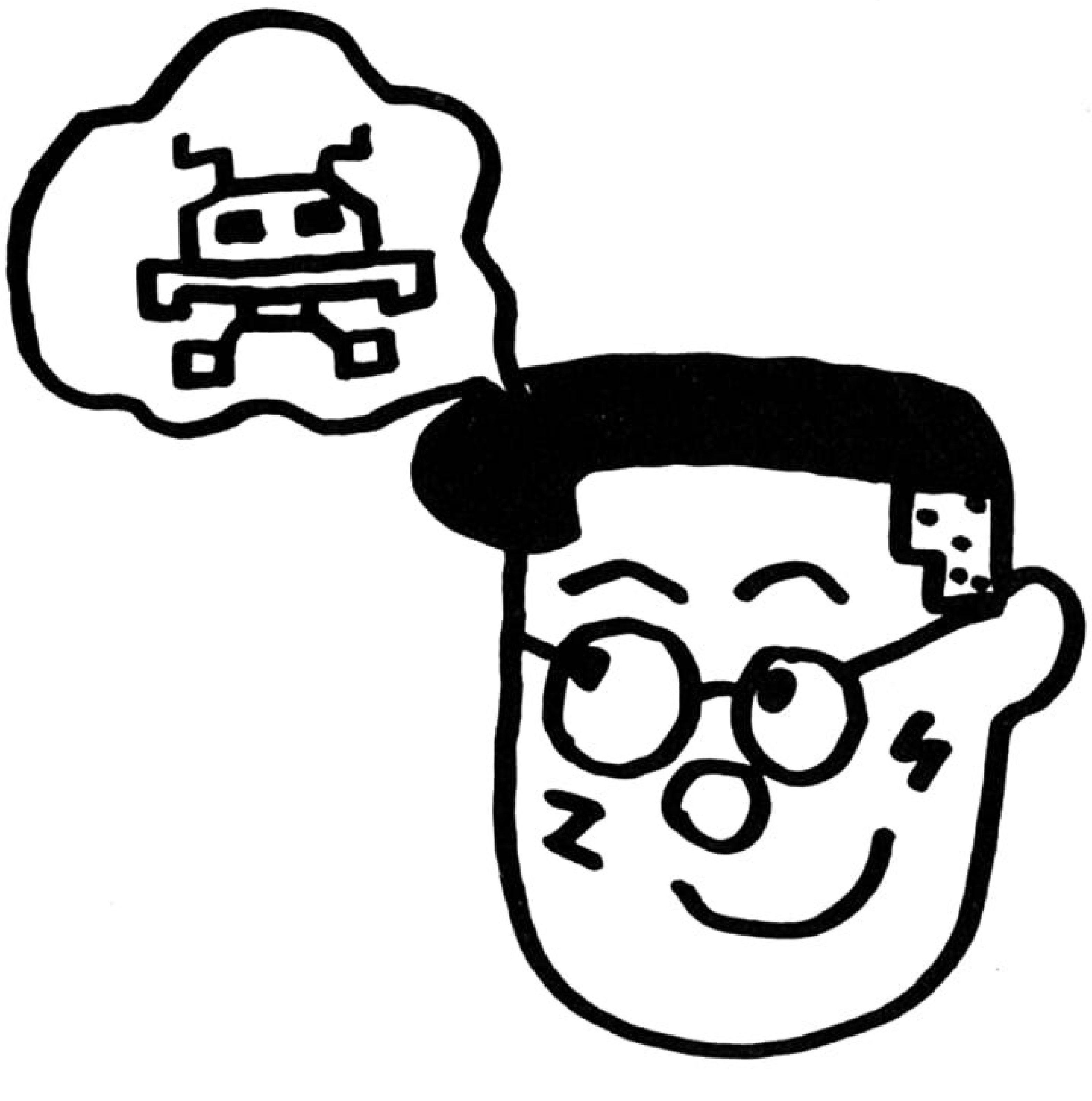
**画面３-４**



GAME CHARACTER

# 4

# ゲームキャラクタ操縦法 PART1



「ゲームキャラクタを動かす」，あるいは「パターンを動かす」と一口にいっても，次の３つの意味があります．

**(1) 表示する位置を変える**
**(2) パターンの形を変える**
**(3) (1)と(2)の組み合わせ**

ここでは，(1)の動きを説明しましょう．(2)，(3)については次のPART２で考えてみます．

# 1 パターンの動かし方

パターンを動かすということは，PUT SPRITE文で表示する位置を変えることです．１つのプレーンには１つのパターンしか表示できませんから，同じプレーンの異なった場所に表示しようとすると，前に表示されていたものは消えてしまいます．そんなスプライトの特徴が，パターンの移動を手軽なものにしているわけです．

そのような動きをするものも，いままでにプログラムとしてすでに出てきています．たとえば，１章のカーソル(**リスト１－２**)，また３章のPUT SPRITE文でSTEPを使った**リスト３－４**(飛行機が飛ぶデモンストレーション)などは，その一つです．

１章のカーソルは，キーボードで入力された命令によって動きが制御されている例です．一方，３章の例はあらかじめ決まった動きしかしません．そこで，実行のたびにさまざまな位置に表示されるパターンと，あらかじめ決まった動きをするパターンの２つにわけて，ゲームの中で動いているものを考えてみましょう．

具体例として，インベーダーゲームを取り上げてみます．インベーダーゲームの中のキャラクタで，動くものを考えてみると，インベーダー，UFO，砲台などがあげられます．この中で，動き方があらかじめ決まっているものはインベーダー，UFOなどで，人間の操作で動くものが砲台ということになります．

このように，１つのゲームの中でもキャラクタごとに分解して考えれば，基本的には１つのスプライト・パターンの動作と考えることができます．では，いったいどうすれば，パターンを動かすことができるのでしょうか．

# 2 インベーダー型の動き

まず，あらかじめ決まった動きを考えてみると，最も簡単な例が3章の飛行機のデモンストレーション(**リスト3－4**)でしょう．この例は，x座標とy座標の値をそれぞれ増減しているだけでした．加減算だけでも，あの飛行機の移動が表わせるのですから，もうひとつ工夫すればもっと変わった動きになるかもしれません．そこで，パターンが画面上を斜めに動き，画面の端でバウンドするような動きを作ってみましょう．

リスト4-1

```
100 COLOR 15,4,7:SCREEN 2,0:CLS
110 SPRITE$(0)=STRING$(2,126)
    +STRING$(4,153)+STRING$(2,126)
120 X=120:Y=90:DX=-1:DY=1
130 PUT SPRITE 0,(X,Y)
140 X=X+DX:Y=Y+DY
150 IF X<10 OR X>245
    THEN X=X-DX:DX=-1*DX
160 IF Y<0 OR Y>184
    THEN Y=Y-DY:DY=-1*DY
170 GOTO 130
```

パターンが画面上を動き回るのがわかっていただけたでしょうか．パターンが動く速さは，変数DX，DYが決定しており，それぞれx方向，y方向への速度の成分となっています．ですから，DX，DYに異なった値を――たとえば，DX＝1：DY＝3を代入しておくと，またひと味違った動きを得られるようになります．

表示位置を定数値だけずらすのでは，あまりおもしろい効果を得られません．画面をおもしろくするためには，**リスト4－1**のループの中にさまざまな条件文を加えることによって可能です．また，その条件に合った絵をグラフィック画面に描けば，次のようなプログラムもできます．

リスト4-2

```
100 COLOR 15,4,7:SCREEN 2,0:CLS
110 SPRITE$(0)=STRING$(2,126)
    +STRING$(4,153)+STRING$(2,126)
120 LINE(0,88)-(79,95),,BF
130 LINE(148,88)-(255,95),,BF
140 X=120:Y=90:DX=-1:DY=1
150 PUT SPRITE 0,(X,Y)
160 X=X+DX:Y=Y+DY
170 IF X<10 OR X>245
    THEN X=X-DX:DX=-1*DX
180 IF Y<0 OR Y>184
    THEN Y=Y-DY:DY=-1*DY
190 IF (X<80 OR X>140)AND
   ((Y=80 AND DY>0)OR(Y=95 AND DY<0))
   THEN Y=Y-DY:DY=-1*DY
200 GOTO 150
```

条件文は，複雑になればなるほどおもしろい画面になりますが，条件文を処理する時間を余分に費すために，スプライト本来の比較的速い動きが損なわれてしまう危険があります．

## 3 関数を盛り込めば…

いままでの例は，さまざまな条件を加えることによって，2つの値が変化していく状態をプログラムで定義したことになりますが，1つの値を与えるともう1つの値が決定するような状態——つまり，関数を定義してパターンを動かしてみると，どうなるでしょう．

**リスト4-3**に示したプログラムは，物理で有名な放物運動の軌跡をスプライト・パターンになぞらせたものです．

リスト4-3

```
100 COLOR 15,4,7:SCREEN 2.0:CLS
110 S$=CHR$(&H7E)+STRING$(6.255)
120 S$=S$+CHR$(&H7E)
```

```
130 SPRITE$(0)=S$
140 DIM DY(60)
150 FOR I=0 TO 60
160  DY(I)=(I-35)^2/5
170 NEXT
180 '
190 FOR X=0 TO 240 STEP 4
200  PUT SPRITE 0,(X,DY(X/4))
210  FOR W=0 TO 10:NEXT
220 NEXT
230 GOTO 190
```

このように関数を用いても，一定の運動をパターンで表現することができます．

**リスト4-3**の例は，物が投げ放たれたときの動きをシミュレーションできたことになりますが，もう一つ関数の例として，三角関数を用いてみましょう．

リスト4-4

```
100 COLOR 15,4,7:SCREEN 2,2:CLS
110 SPRITE$(0)=STRING$(16,&HC0)
120 DIM D(63):PI=3.14159
130 FOR I=0 TO 63
140  D(I)=70*SIN(PI*I/15)+90
150  PSET(I*4+10,D(I)+8)
160 NEXT
170 FOR I=0 TO 58
180  FOR J=0 TO 5
190   X=(I+J)*4+10:Y=D(I+J)
200   PUT SPRITE J,(X,Y),,0
210  NEXT
220 NEXT
230 GOTO 170
```

SIN関数の描く軌跡を，同じスプライト・パターンで，5つ同時に表示しながらなぞらせてみました．パターンは5つですが，同じ形のものなので，1つのパターンを表示させた例として取り上げました．なんとなく，ジェット・コースターのように見えませんか？

**リスト4-3，4-4**の例のように，関数演算によってパターンの動きを決定す

る場合には演算しながら表示するよりも，さきに演算結果を配列にたくわえておき，表示の際にはそれらを参照するだけにすれば，動きも多少速くなります（このため，以下のプログラムでは実行後画面に表示されるまで少し待たされます）．これが**リスト4－4**の例のように，繰り返し動作が含まれていればなおさらです．

さて，関数による軌跡ということで，1つの値(x座標)を与えると，もう1つの値(y座標)が決定するものを考えてきましたが，1つの値を与えるとx，y両座標の値が決定するような例も考えられます．

## リスト4-5

```
100 COLOR 15,4,7:SCREEN 2,0:CLS
110 SPRITE$(0)=STRING$(8,255)
120 DIM DX(44),DY(44)
130 PI=3.14159
140 FOR I=0 TO 44
150  DX(I)=80*COS(PI*I*4/90)+130
160  DY(I)=80*SIN(PI*I*4/90)+100
170 NEXT
180 '
190 FOR I=0 TO 44
200  PUT SPRITE 0,(DX(I),DY(I))
210  FOR W=0 TO 10:NEXT
220 NEXT
230 GOTO 190
```

x，y両座標を決定するために与える値のことを媒介変数といいます．むずかしい話はべつにして，**リスト4－5**のような例は，非常に複雑な動き――それも，決まった動きを与えてくれます．次の例もまたその典型でしょう．

## リスト4-6

```
100 COLOR 15,4,7:SCREEN 2,0:CLS
110 SPRITE$(0)=STRING$(8,255)
120 DIM DX(89),DY(89)
130 PI=3.14159
140 FOR I=0 TO 89
150  DX(I)=80*COS(PI*I*4/90)+130
```

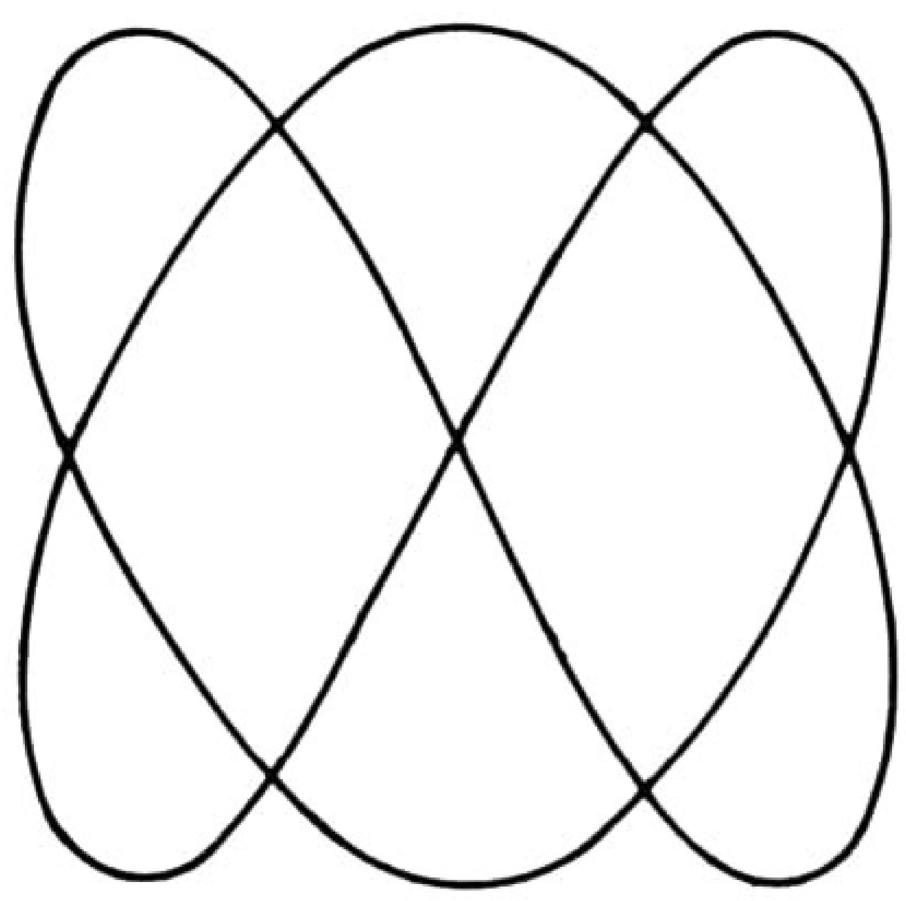
```
160  DY(I)=80*COS(PI*I*6/90+PI/4)+100
170 NEXT
180 '
190 FOR I=0 TO 89
200  PUT SPRITE 0,(DX(I),DY(I))
210  FOR W=0 TO 10:NEXT
220 NEXT
230 GOTO 190
```

これは，リサージュ曲線とか，単にリサージュとかいわれるものです．**リスト 4－6** の行番号140からはじまるFORループで，スプライト・パターンの動く点の座標をまとめて計算しています．ここで計算された点は，**図 4－1** のような図形(これがリサージュ）上の各点の座標です．画面上の各点を表示させるために，行番号170のNEXTの直前に，

165 PSET(DX(I),DY(I))

を挿入すれば，リサージュ図形が**図 4－1** と同じであることがよくわかると思います．

**図 4－1**



複雑な関数にしたがった動きは，表示(移動)のスピード・アップのために，演算しながら表示するのでなく，さきに表示位置の演算をまとめて処理し，結果を配列に保存しておいて，その配列を参照するだけで表示できるようにしておきます．この方法は，よく考えると，複雑な関数で得た結果を配列にしまっておかなくても，おもしろい動きをする自分で作った座標値を，配列の値としてもよいことになります．

# 4 影にかくれるスプライト・パターン

PUT SPRITE文には，座標の値として，特別な機能をもった値が用意されています．この値を配列の内容として使用すれば，条件文を減らす手助けともなります．

用意されている特別な値は，「208」と「209」で，両方ともy座標の値として指定したときに，機能するようになっています．それぞれの意味は，次のとおりです．

**208：指定したスプライト・プレーン以降のスプライト・パターンがすべて消える．**

**209：指定したスプライト・プレーンに表示されていたスプライト・パターンを消す．**

ある範囲だけを表示したくない場合には，この機能を使えば，表示のときにIF文などの条件をつけてその判定をする必要はなくなります．あらかじめ配列に用意した座標値のうち，Y座標を「209」に指定しておいた範囲については，その間スプライト・パターンが表示されないことになります．

リスト4-7

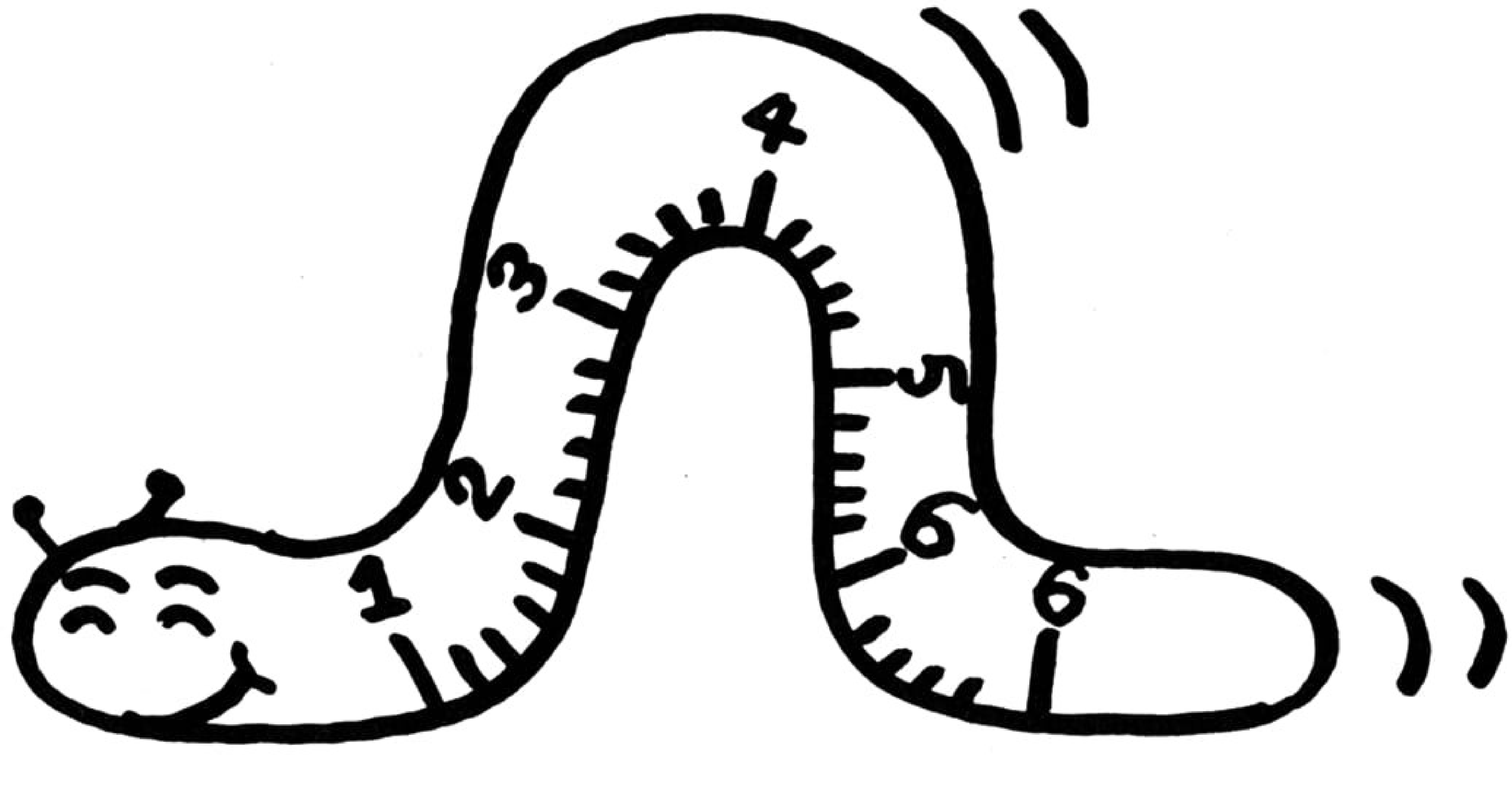
```
100 COLOR 15,1,1:SCREEN 2,0:CLS
110 SPRITE$(0)=STRING$(8,255)
120 DIM DX(89),DY(89)
130 PI=3.14159
140 CIRCLE(130,100),60
150 PAINT(130,100)
160 '
170 FOR I=0 TO 89
180  DX(I)=80*COS(PI*I*2/90)+130
190  DY(I)=80*COS(PI*I*2/90+PI/4)+100
200 NEXT
210 '
220 FOR I=11 TO 24:DY(I)=209:NEXT
230 '
240 FOR I=0 TO 89
250  PUT SPRITE 0,(DX(I),DY(I)),9
260  FOR W=0 TO 10:NEXT
```

```
270 NEXT
280 GOTO 240
```

**リスト4—7**でも，CIRCLE文で円を描いて塗りつぶすだけでなく，もっと凝って大陸や海などを書き込めば，地球のまわりを飛ぶ人工衛星のように見えたかもしれません．

行番号220のFORループで，添字が「11」から「24」の範囲の配列DYに「209」を代入しています．したがって，この範囲の添字が指定されたときにはスプライト・パターンは表示されません．

もし，この機能がなければ，行番号240からのFORループ(表示のためのループ)の中で，添字の範囲をチェックして，「11」から「24」の間の値なら表示しないように処理しなければなりません．その意味でも，配列に表示位置を持っていて，このような機能を用いることができるのは，画面上でスプライト・パターンに一定の動きをさせるのに便利だと思われます．



# 5 砲台型の動き(キー操作による移動)

あらかじめ決まっていない動きについて考えてみると，最も顕著な例は，座標の値を乱数によって与えるものでしょう．もう1つ例として考えられるのは，人間がキーボードで入力した状態によって移動するものでしょう．すでに出てきた1章のカーソルなどは，その例といえます．たとえば，インベーダーゲームの砲台の動きは，次のようなプログラムで実現できます．

リスト4-8

```
100 COLOR 15,4,7:SCREEN 2,0:CLS
110 GOSUB 260
120 X=120:DX=5
130 PUT SPRITE 0,(X,183)
140 CH=ASC(INPUT$(1))
150 IF CH=13 THEN 240
160  IF CH=28 THEN X=X+DX
170  IF CH=29 THEN X=X-DX
180  IF X>247 THEN X=X-DX
190  IF X<0   THEN X=X+DX
200  PUT SPRITE 0,(X,183)
210  CH=ASC(INPUT$(1))
220 GOTO 150
230 '
240 END
250 '
260 FOR I=1 TO 8
270  READ D$:D=VAL("&B"+D$)
280  S$=S$+CHR$(D)
290 NEXT
300 SPRITE$(0)=S$
310 RETURN
320 '
330 DATA 00000000
340 DATA 00000000
350 DATA 00011000
360 DATA 00011000
370 DATA 01111110
380 DATA 11111111
```

```
390 DATA 11111111
400 DATA 11111111
```

←と→キーを用いて，砲台(画面上)を左右に移動させるプログラムが，**リスト4－8**です．リターンキーでプログラムは終了します．

この例は，y座標が固定されていますが，横方向の位置はキーボードの入力によって変化するので，決まった動きではありません．**リスト4－8**に，次の**リスト4－9**を加えれば，さらにリアルな砲台のシミュレーションとなります．

## リスト4-9

```
115 SPRITE$(1)=STRING$(8,&H18)
116 ON STRIG GOSUB 510
117 STRIG(0) ON
500 '
510 STRIG(0) OFF
520 FOR Y=183 TO -10 STEP -3
530  PUT SPRITE 1,(X,Y),9
540 NEXT
550 STRIG(0) ON
560 RETURN
```

**リスト4－9**をつけ加えることによって，スペースバーを押したときにビームが砲台から発射されます．特別な処理をしていないので，ビームが画面の上に消えるまでの間，砲台は移動できません．さらに音をつけたりすると，インベーダーゲームの一部ができてしまったことになります．



# 6 ヴァリエーションのある動き

さて，まったくのランダムな動きでなく，ある動きのパターンをいくつかあらかじめ用意しておいて，それを選ぶような場合も考えられます．ちょうど，いままでに出てきた，「一定の動き」と「ランダムな動き」の組み合わせと考えられます．組み合わせをユーザーが決定できるように，ファンクションキーの割込みを用いて，プログラムを作ってみましょう．

リスト4-10

```
100 COLOR 15,4,7:SCREEN 2,0:CLS
110 GOSUB 290
120 ON KEY GOSUB 230,240,250,260,270
130 FOR I=1 TO 5:KEY(I) ON:NEXT
140 DX=3:DY=1:X=120:Y=90
150 '
160 PUT SPRITE 0,(X,Y)
170 X=X+DX:Y=Y+DY
180 IF X<0 OR X>248
    THEN X=X-DX:DX=-1*DX
190 IF Y<0 OR Y>186
    THEN Y=Y-DY:DY=-1*DY
200 FOR W=0 TO 30:NEXT
210 GOTO 160
220 '
230 DX=5*SGN(DX):DY=SGN(DY):RETURN
240 DX=SGN(DX):DY=5*SGN(DY):RETURN
250 DX=SGN(DX):DY=SGN(DY):RETURN
260 DX=-5*SGN(DX):RETURN
270 DY=-5*SGN(DY):RETURN
280 '
290 FOR I=1 TO 8:READ D$
300  D=VAL("&H"+D$):S$=S$+CHR$(D)
310 NEXT
320 SPRITE$(0)=S$
330 RETURN
340 '
350 DATA 42,DB,99,FF,FF,99,DB,42
```

このプログラムでは，5つのファンクションキーを適当に押すと，動きが変わります．

ゲームの中には，動きを組み合わせるだけで十分成り立っているものが数多くあります．かえって，まったくのでたらめな動きという例のほうが少ないと思います．したがってリスト4-10は，ゲームの基本的な部分をつけ加えると，バラエティに富んだものができる例ともいえるでしょう．

いままで，さまざまな動き方を見てきましたが，この章のまとめとして『アミダくじ』のプログラムを作ってみましょう．『アミダくじ』は，「あらかじめ用意されたコース」になっていますが，「くじ」のどこを引くかによってそのコースがさまざまに変化するので，完全に決まった動きとはいえません．

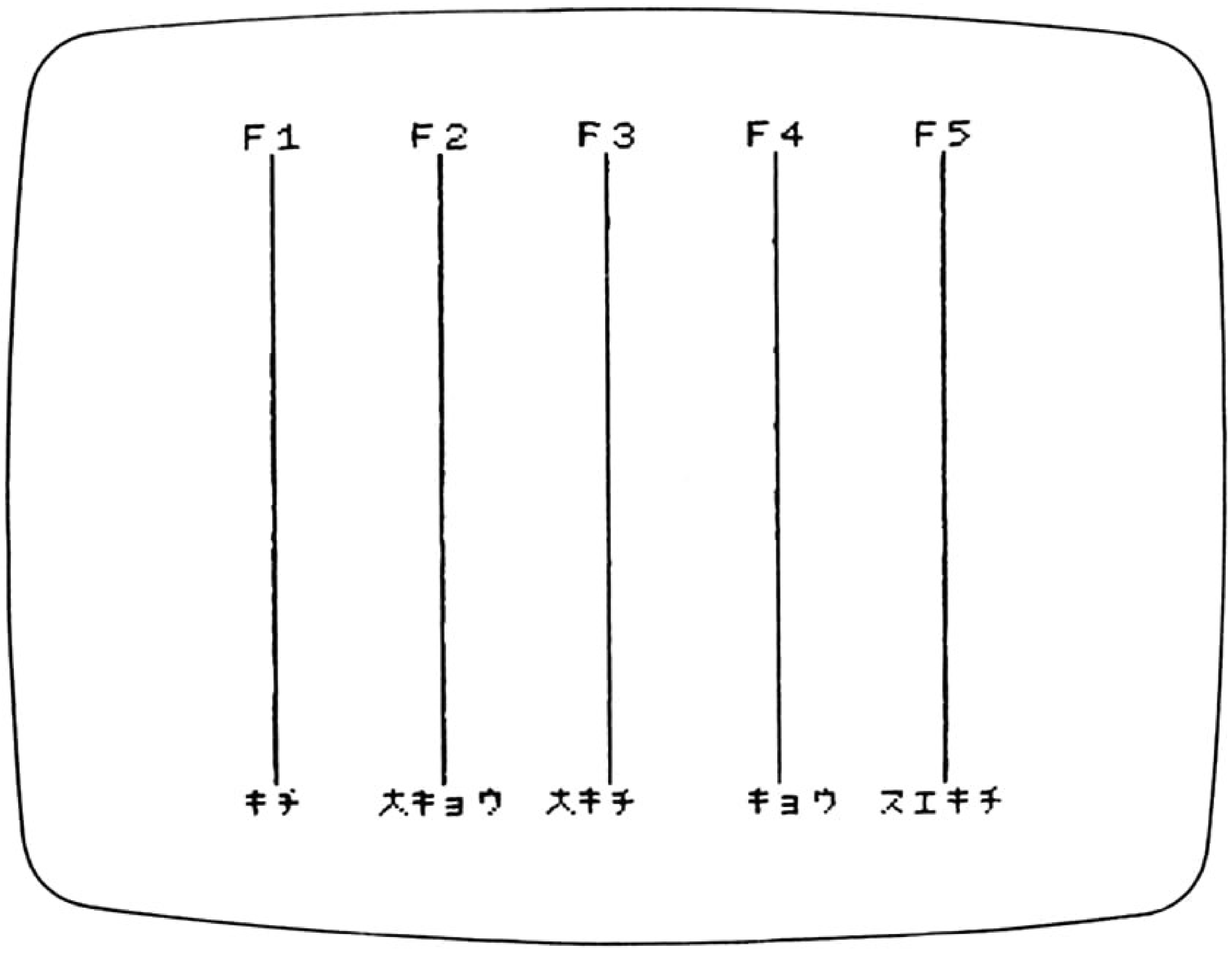
リスト4-11

```
100 COLOR 15,4,7:SCREEN 2,0:CLS
110 SPRITE$(0)=STRING$(8,255)
130 FOR I=1 TO 5:KEY(I) ON:NEXT
140 DEF FNN$(X$)=RIGHT$(X$,LEN(X$)-1)
150 DIM BR(4,8)
160 '
170 FOR J=0 TO 8
180  FOR I=0 TO 3
190   D=10*RND(-TIME)
200   IF BR(I,J)=-1 THEN 230
210   IF D<4 THEN BR(I,J)=0:GOTO 230
220    BR(I,J)=1:BR(I+1,J)=-1
230  NEXT
240 NEXT
250 '
260 OPEN "GRP:" FOR OUTPUT AS #1
270 FOR I=0 TO 4
280  X=I*40+40
290  PRESET(X-6,2)
300  PRINT #1,"F";FNN$(STR$(I+1))
310  LINE(X,10)-(X,180)
320  PRESET(X-15,182)
330  READ A$:PRINT #1,A$
340 NEXT
350 CLOSE #1
360 '
```

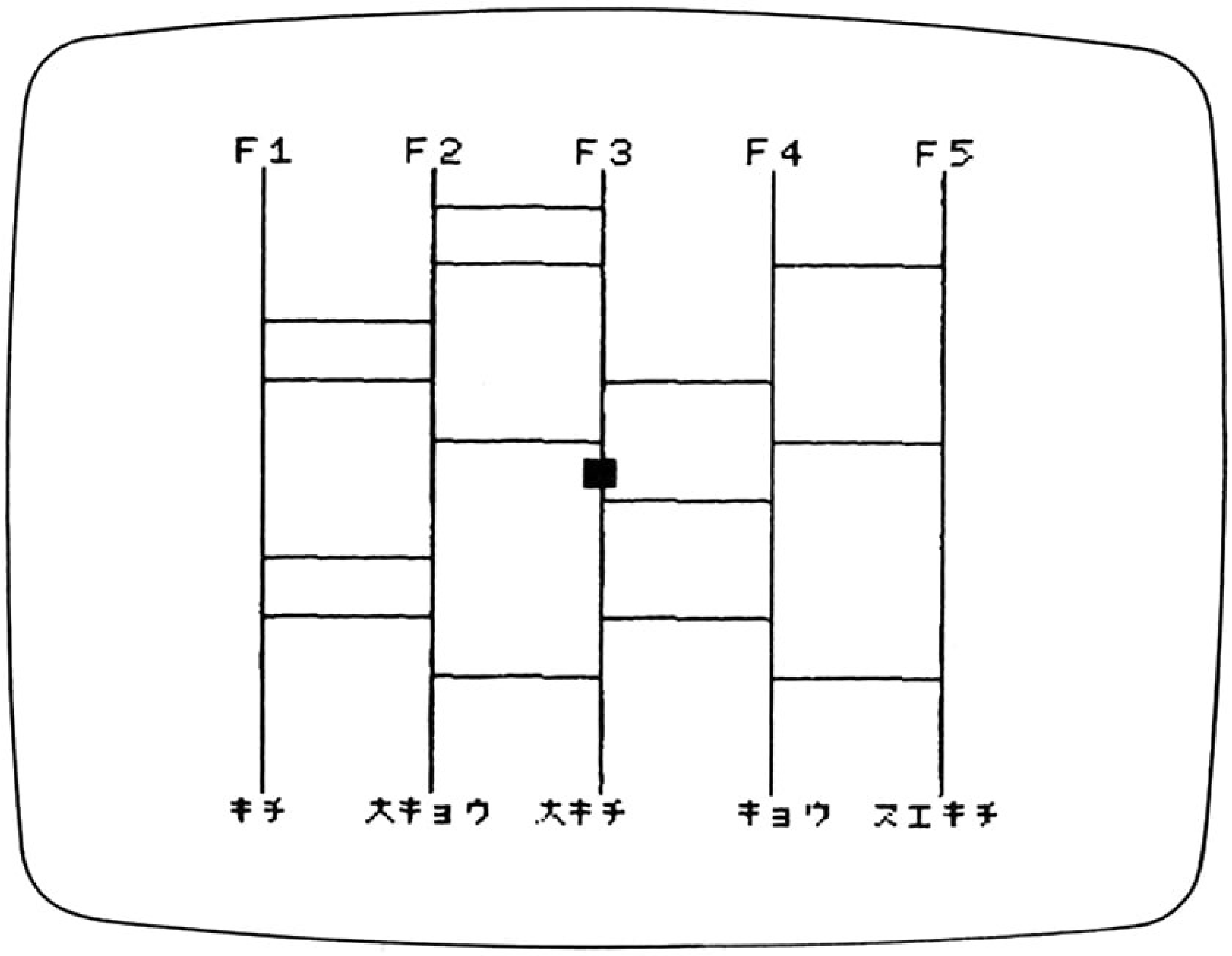
```
365 ON KEY GOSUB 770,780,790,800,810
370 GOTO 370'WAIT FOR INTERRUPT
380 '
390 PUT SPRITE 0,(A*40+36,1),9
400 FOR J=0 TO 8
410  FOR I=0 TO 3
420   IF BR(I,J)<>1 THEN 450
430    X=I*40+40:Y=J*16+20
440    LINE(X,Y)-STEP(40,0)
450  NEXT
460 NEXT
470 '
480 X=A*40+36
490 FOR Y=1 TO 15
500  PUT SPRITE 0,(X,Y),9
510  GOSUB 880
520 NEXT
530 '
540 FOR B=0 TO 8
550  Y=B*16+16
560  IF BR(A,B)=0 THEN 630
570   S=A*40+36:E=(A+BR(A,B))*40+36
580   FOR M=S TO E STEP BR(A,B)
590    PUT SPRITE 0,(M,Y),9
600    GOSUB 880
610   NEXT
620 '
630  A=A+BR(A,B):X=A*40+36
640  S=B*16+16:E=(B+1)*16+15
650  FOR M=S TO E
660   PUT SPRITE 0,(X,M),9
670   GOSUB 880
680  NEXT
690 NEXT
700 '
710 FOR M=161 TO 183
720  PUT SPRITE 0,(X,M),9
730  GOSUB 880
740 NEXT
750 GOTO 750
760 '
770 A=0:GOTO 830
780 A=1:GOTO 830
790 A=2:GOTO 830
```

```
800 A=3:GOTO 830
810 A=4
820 '
830 FOR I=1 TO 5
840  KEY(I) OFF
850 NEXT
860 RETURN 390
870 '
880 FOR W=0 TO 10:NEXT:RETURN
890 '
900 DATA " キチ","大キョウ","大キチ"
910 DATA " キョウ","スエキチ"
```

画面4－1




画面4－2



プログラムをRUNさせると，しばらくして，**画面4－1**のような表示が現われます．次に，[F1]から[F5]を1つ選んで押してください．すると横線が現われて，スプライト・パターンが『アミダくじ』の上をなぞっていきます(**画面4－2**参照)．出てくる横線は毎回違う位置に現われます． ゴールに着くとそのまま停止しますので，[CTRL]＋[STOP]でプログラムを止めてください．

以下にプログラムの簡単な説明を加えておきます．

| | |
|---|---|
| 配列BR（，） | ：各縦線の分岐情報を持っている．分岐しうる点での値は，0＝分岐しない，1＝右に，－1＝左に分岐する，という意味を持っている． |
| 行番号170～240 | ：分岐情報を作る． |
| 行番号260～350 | ：縦線と文字の表示． |
| 行番号370 | ：割り込みを待つ． |
| 行番号390～460 | ：横線を，分岐情報をもとにして表示． |
| 行番号480～740 | ：アミダくじをなぞる． |
| 行番号770～860 | ：割り込み処理． |
| 行番号880 | ：ディレイ・ルーチン． |



GAME CHARACTER

# 5

# ゲームキャラクタ操縦法 PART2



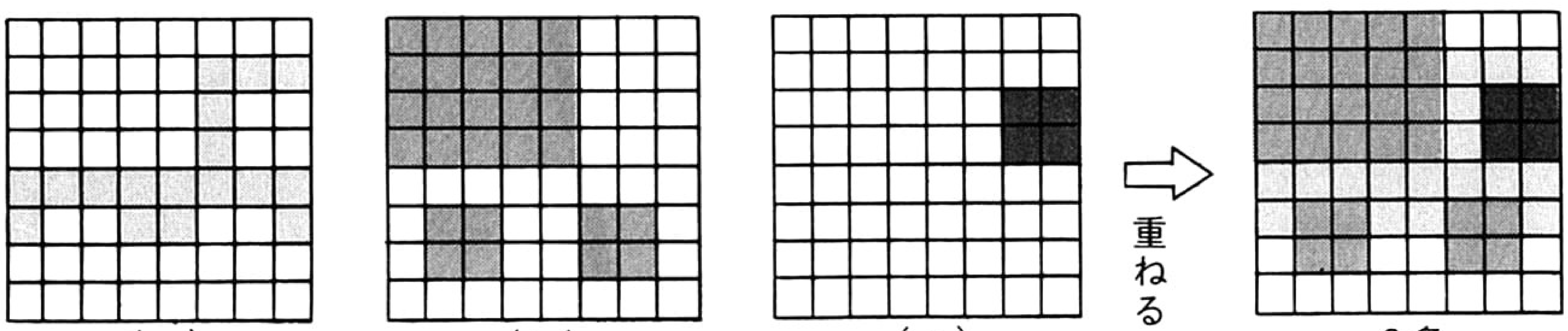
1つのパターンの動かし方は，PART 1でひと通り述べてきました．ここでは，複数のパターンを動かすことをお話するわけですが，基本は1つのパターンを動かすことと変わりません．前章と多少変わる部分をピック・アップするのがここでの目的ですから，PART 1で述べてあることはあまり詳しくふれません．

# 1 キャラクタの色づけ

前章の最初に述べましたが，「パターンを動かす」という意味は3つあります．ここでは複数のパターンを用いるので，「パターンの形を変える」とか，「形を変えながら移動する」といった動き方を考えます．さらに，複数のパターンを用いることによって，パターンに色(複数の色)をつけてみるということも考えられます．

では，まず1つのパターンに色をつけることを考えてみましょう．
前々からいっているように，水平方向にはスプライト・パターンを5つ以上並べて表示できません．複数のパターンを用いて，1つのパターンに色をつけるためには，形の異なったいくつかのパターンに色をつけて，重ねて表示しなければなりません(**図5-1**参照)．「重ねる」ということは，水平方向に並べることと同じですから，結局1つのパターンに用いることのできる色は，最高4色にかぎられます．ここでは，3色の例を見てみましょう．

**図5-1**




## リスト5-1

```
100 COLOR 15,1,1:SCREEN 2,1:CLS
110 GOSUB 210
120 LINE(0,120)-(255,191),9,BF
130 FOR X=5 TO 250 STEP 2
```

```
140  PUT SPRITE 0,(X,104),4
150  PUT SPRITE 1,(X,104),10
160  PUT SPRITE 2,(X,104),7
170  FOR W=0 TO 20:NEXT
180 NEXT
190 GOTO 130
200 '
210 FOR J=0 TO 2:S$=""
220  FOR I=1 TO 8
230   READ D$:D=VAL("&H"+D$)
240   S$=S$+CHR$(D)
250  NEXT
260  SPRITE$(J)=S$
270 NEXT
280 RETURN
290 '
300 DATA 00,07,04,04,FF,99,00,00
310 DATA F8,F8,F8,F8,00,66,66,00
320 DATA 00,00,33,33,00,00,00,00
```

パターンの動かし方は，各色のパターンを１つずつ順に移動させるだけなので，まったく特別なことはありません．しかし，１つのパターンを動かすときよりも，動きが少しだけ遅く感じられるかもしれません．その場合には，移動の際に使用していた，ディレイ・ルーチンのFOR文のループ（170行）の上限値を小さくしたり，ディレイ・ルーチンを取り去ってしまえば，十分解決がつくはずです．

複数色のパターンを用いたときに注意しなければいけないのは，複数色のパターンを２つ以上表示したとき，それらが水平方向に並ばないようにすることです．しかし，１つのパターンに２色しか使っていないのであれば，２つまでなら並べて表示できます．もちろん，この場合も横に５つ以上のパターンが並ばないことは，念頭においておかなければなりません．

## 2 複数のパターンをあやつる

スプライト・パターンは，画面に表示したときに，あまり大きなものではないので，複数色にしてもそのメリットには限界があります．ですから，多くのパターンを使う場合には，１つのパターンに複数の色をつける目的よりも，異なった

パターンをいくつか用意して，それらを組み合わせて動かしたほうがおもしろいことができそうです．

たとえば，ポピュラーなゲームを考えてみると，インベーダーゲームなどのように，複数のパターンを動かすもののほうがはるかに多いのです．最近のゲームはかなり複雑でカラフルになってきていますが，MSXには，インベーダーゲームのような複数のパターンを扱うゲームのほうがどちらかというと好都合です．

すでに，1章のはじめに出てきた，デモンストレーションのプログラムなどはそのよい例でしょう．1章の例はパターンが3つありましたが，パターンが2つの場合を見てみましょう．

## リスト5-2

```
100 COLOR 15,1,1:SCREEN 2,1:CLS
110 GOSUB 260
120 X=100:Y=90:A=140:B=90
130 DX=-2:DY=-3:DA=3:DB=2
140 C0=5:C1=9
150 '
160 PUT SPRITE 0,(X,Y),C0
170 PUT SPRITE 1,(A,B),C1
180 X=X+DX:Y=Y+DY
190 IF X<5 OR X>236 THEN
     X=X-DX:DX=-1*DX
200 IF Y<0 OR Y>175 THEN
     Y=Y-DY:DY=-1*DY
210 A=A+DA:B=B+DB
220 IF A<5 OR A>236 THEN
     A=A-DA:DA=-1*DA
230 IF B<0 OR B>175 THEN
     B=B-DB:DB=-1*DB
240 GOTO 160
250 '
260 FOR I=1 TO 8
270  READ P$,Q$
280  P=VAL("&B"+P$):Q=VAL("&B"+Q$)
290  S0$=S0$+CHR$(P)
300  S1$=S1$+CHR$(Q)
310 NEXT
320 SPRITE$(0)=S0$
```

```
330 SPRITE$(1)=S1$
340 RETURN
350 '
360 DATA 00010000,00111000
370 DATA 00111000,01000100
380 DATA 01010100,01000100
390 DATA 10010010,00111000
400 DATA 00111000,00010000
410 DATA 01000100,11111110
420 DATA 01000100,00010000
430 DATA 00111000,00010000
```

2つのパターンを動かすときには，単に表示位置を決めている変数を，パターンごとに用意すればいいだけです．したがって，1つのパターンを動かすときと内容的には，まったく同じことを行なっているのです．

配列を利用する場合も同じです．各パターンごとに配列を用意すれば，各パターンごとに動きを定義することができます．しかし，パターンの軌跡が，同じ場合には，そのようなことはせずに，用意する配列は1つですみます．

## リスト5-3

```
100 COLOR 15,4,7:SCREEN 2,0:CLS
110 SPRITE$(0)=STRING$(8,255)
120 SPRITE$(1)=STRING$(8,255)
130 DIM DX(89),DY(89)
140 CIRCLE (130,100),60
150 PAINT (130,100)
160 '
170 PI=3.14159
180 FOR I=0 TO 89
190  DX(I)=80*COS(PI*I*2/90)+130
200  DY(I)=80*COS(PI*I*2/90+PI/4)+100
210 NEXT
220 '
230 FOR I=11 TO 24:DY(I)=209:NEXT
240 '
250 FOR I=0 TO 89
260  PUT SPRITE 0,(DX(I),DY(I)),3
270  IF I<45 THEN J=I+45 ELSE J=I-45
280  PUT SPRITE 1,(DX(J),DY(J)),9
```

```
290  FOR W=0 TO 10:NEXT
300 NEXT
310 GOTO 250
```

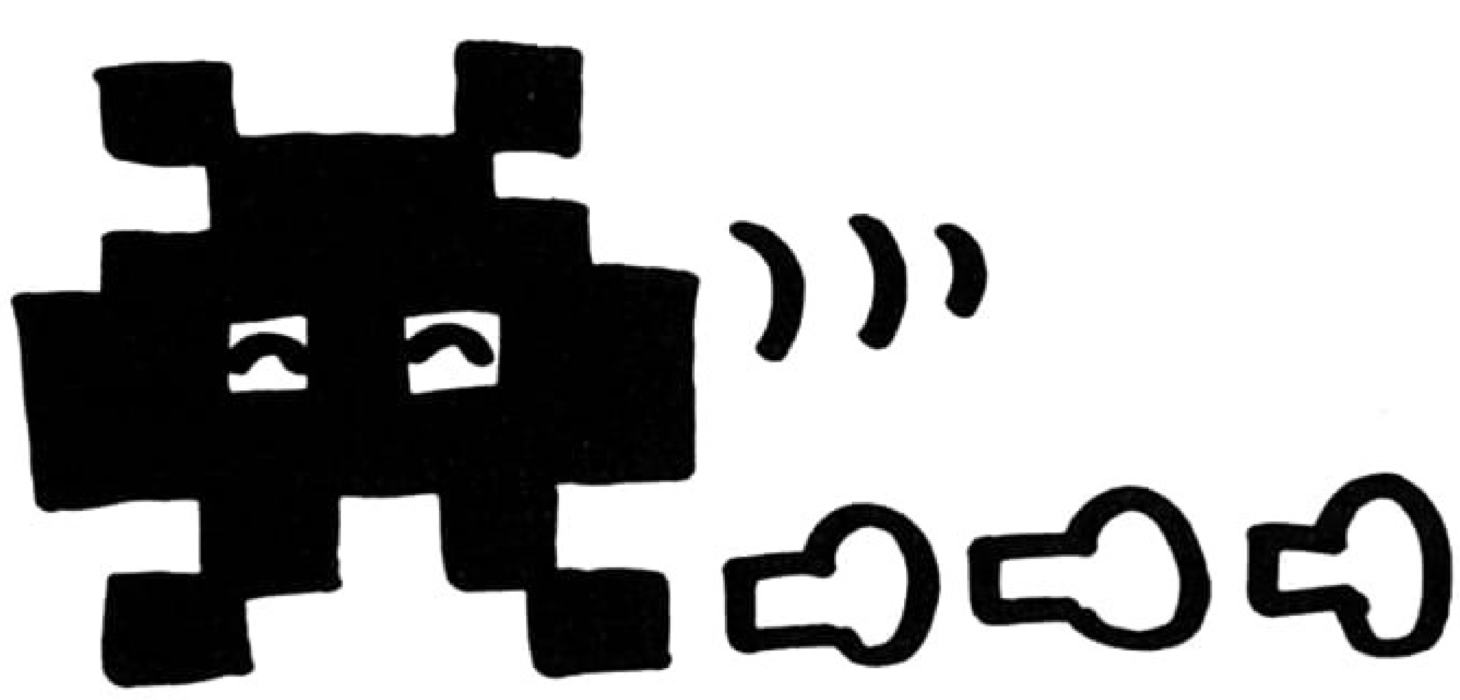
上の例は，**リスト 4－7** と，プログラムの構造がまったく同じです．用意してある配列も１つのパターン分しかありません．そのかわり，２つのパターンで参照する配列の要素がかならず異なっています．つまり，行番号270の条件文で，異なるところを参照するように操作しているのです．したがって，１つのパターン用の配列でも２つのパターンを表示して動かすことが可能なのです．

**リスト 5－3** の例は，２つのパターンで１つの配列なのですが，２つのパターンの通るコースは同じです．そこで，行番号260を次のように変えてみてください．

```
260 PUT SPRITE 0, (260-DX(I), DY(I)), 3
```

このようになおして実行すると，あたかも，２つの異なった軌道のために２つの配列を作っているようです．しかし，プログラムを入力したみなさんは，１つのパターンのためにしか配列を用意していないことを知っているわけです．１つのパターンのための座標データでも，十分におもしろい動きが作れることがわかったでしょうか？

配列を使用するメリットは，パターンの軌跡の座標を配列に持ってさえいれば，**リスト 5－3** のようなプログラムに移すことができることなのです．



# 3 インベーダーのデモンストレーション

1つのパターンから，2つのパターンへと順に考えてきましたが，2つのパターンの扱いさえわかれば，そこからさきはいくつのパターンでも考え方は同じです．PART 1で砲台とビームのデモンストレーションを作ったので，ここではインベーダーのデモンストレーションを作ってみましょう．

リスト5-4

```
100 COLOR 15,4,7:SCREEN 2,0:CLS
110 GOSUB 300
120 X=200:Y=20:DX=-2
130 IF Y>150 THEN 190
140 GOSUB 210
150 X=X+DX
160 IF X<10 OR X>200
    THEN DX=-1*DX:X=X+DX:Y=Y+10
170 GOTO 130
180 '
190 END
200 '
210 FOR J=0 TO 3
220  FOR I=0 TO 3
230   P=I+J*4
240   PUT SPRITE P,(X+I*12,Y+J*10),9,J
250   FOR W=0 TO 10:NEXT
260  NEXT
270 NEXT
280 RETURN
290 '
300 FOR I=0 TO 3 STEP 2:S$=""
310  FOR J=1 TO 8
320   READ D$:D=VAL("&H"+D$)
330   S$=S$+CHR$(D)
340  NEXT
350  SPRITE$(I)=S$:SPRITE$(I+1)=S$
360 NEXT
370 RETURN
380 '
390 DATA 10,38,7C,D6,FE,28,28,44
```

```
400 DATA 28,7C,FE,BA,FE,FE,28,44
```

行番号210から始まるサブルーチンが，16匹のインベーダーを表示させる箇所です．変数XとYには，16匹のインベーダーを1ブロックと考えたときの左上の座標を示しています．

行番号300からのサブルーチンは，インベーダーのパターンをスプライト・パターンとして定義している部分です．ここでは2種類のインベーダーを用意しました(**図5－2**参照)．

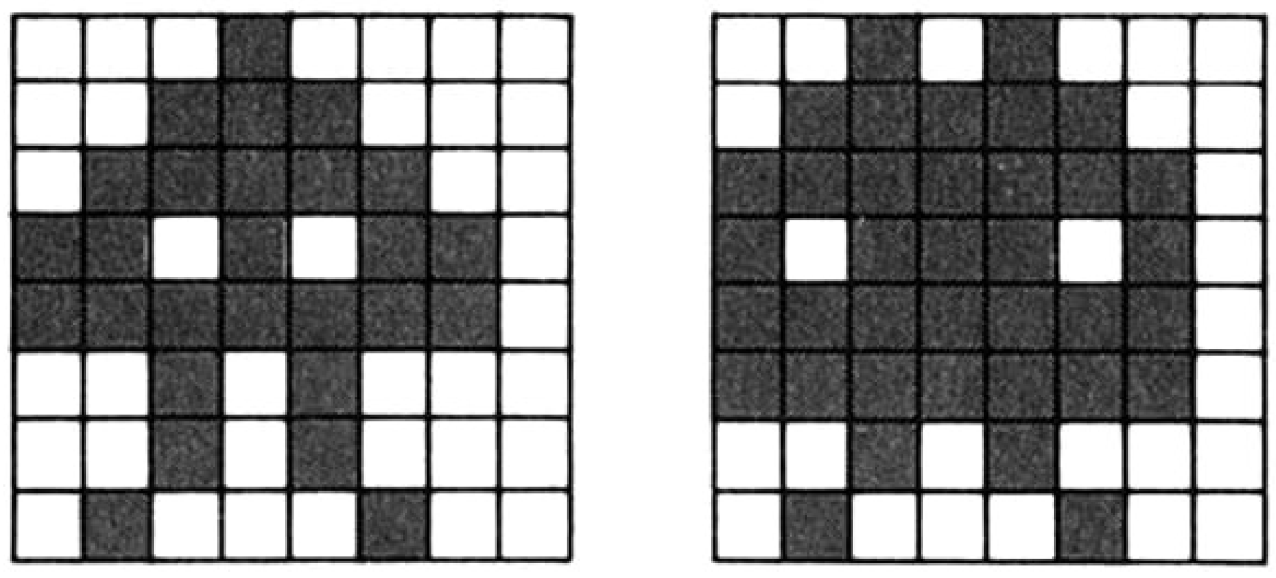


図5－2

このプログラムと，**リスト4－8**および**4－9**の砲台とビームのデモンストレーションのプログラムをうまく組み合わせれば，ほとんどインベーダーゲームができ上がってしまったようなものです．

# 4 アニメーションへの応用

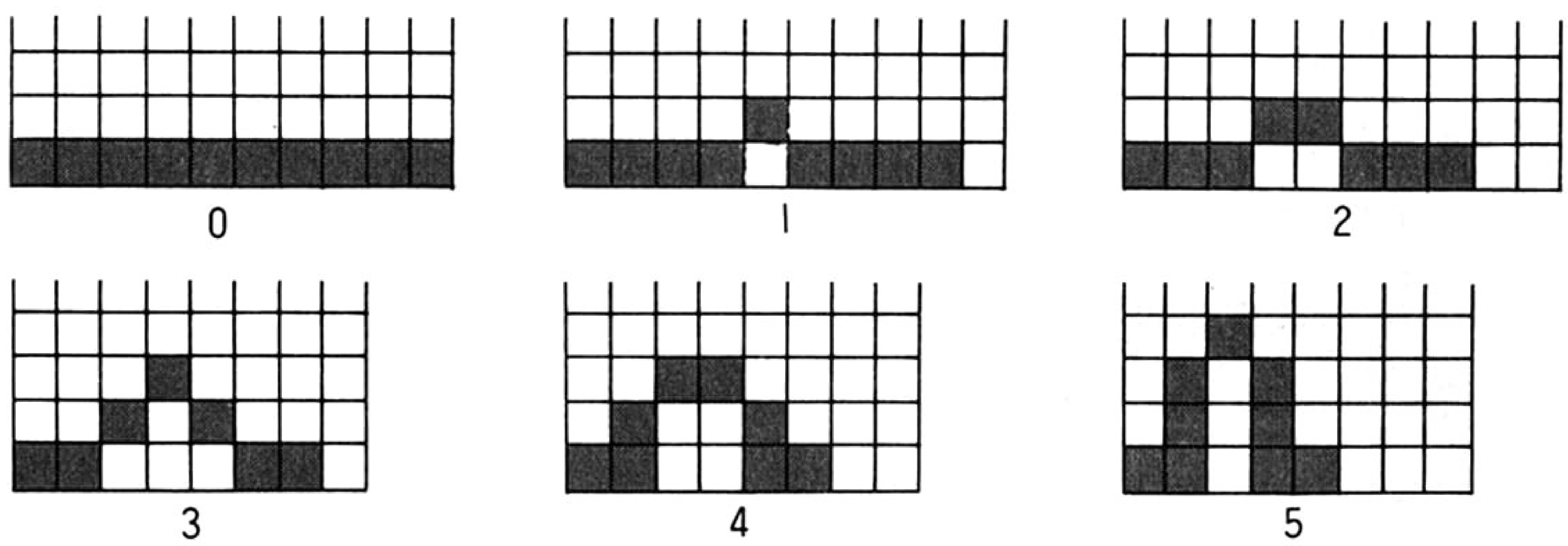
複数のパターンの位置が変化するだけの例を見てきましたが，パターンが形を変化させながら移動する例も考えられます．

リスト5-5

```
100 COLOR 15,1,1:SCREEN 2,1:CLS
110 GOSUB 300
120 X=10
130 IF X>200 THEN 230
140 FOR I=0 TO 5
150  GOSUB 250
```

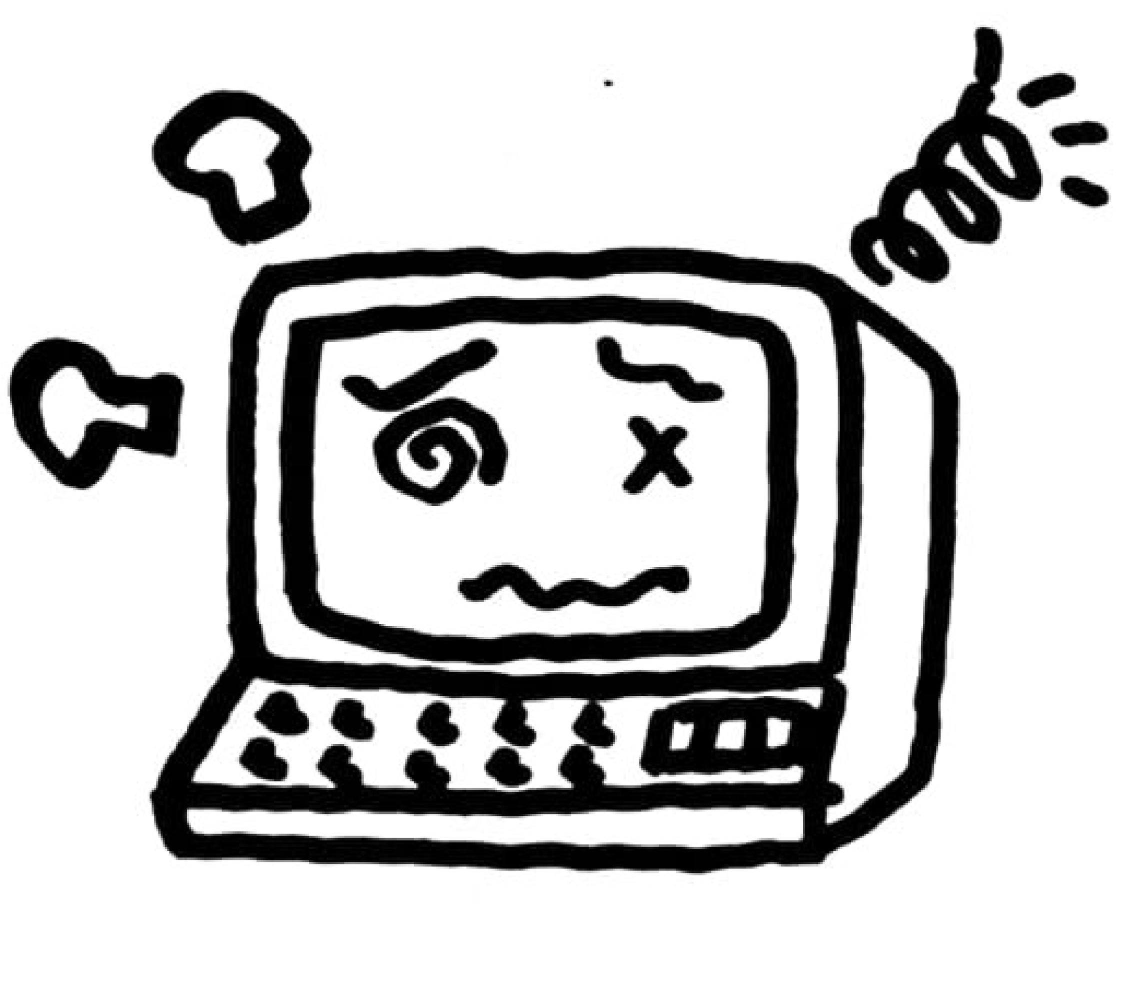
```
160  X=X+1
170 NEXT
180 FOR I=5 TO 1 STEP -1
190  GOSUB 250
200 NEXT
210 GOTO 130
220 '.
230 END
240 '
250 PUT SPRITE 0,(X,90),3,I
260 IF I=0 OR I=1
    THEN PUT SPRITE 1,(X+8,90),3,I+6
   ELSE PUT SPRITE 1,(0,209)
270 FOR W=0 TO 20:NEXT
280 RETURN
290 '
300 FOR J=0 TO 7:S$=STRING$(4,0)
310  FOR I=1 TO 4
320   READ D$:D=VAL("&H"+D$)
330   S$=S$+CHR$(D)
340  NEXT
350  SPRITE$(J)=S$
360 NEXT
370 RETURN
380 '
390 DATA 00,00,00,FF
400 DATA 00,00,08,F7
410 DATA 00,00,18,E7
420 DATA 00,10,28,C6
430 DATA 00,30,48,CC
440 DATA 20,50,50,D8
450 DATA 00,00,00,C0
460 DATA 00,00,00,80
```

図5－3



**リスト5－5**は,「尺とり虫」が画面を横切るようなパターンを作ってみました.このアニメーションをコマ数にわけると6コマで,その内容は**図5－3**のようになっています.これをそれぞれ「0」から「5」までのスプライト・パターンに振りわけて,0→1→2→3→4→5→4→3→2→1という順で，繰り返し表示していきます．**図5－3**を見て気づいた人もいると思いますが，横が8ドット以上あるのに，定義しているスプライト・パターンは8×8ドットです．このはみ出た部分は，スプライト・パターンの「6」，「7」に登録してあり，表示のときにだけこれをつぎたしています．

以上のように，多数のパターンを用いれば変わった画面を作れます．データ文を用意するのはたいへんですが，すてきなアニメーションはその苦労を忘れさせてくれるのではないでしょうか．



# 6
# データ作りを簡単に



# 1 見ながらキャラクタが作れる

最もやりやすいデータの作り方は，この本でもいくつか用いられている，DATA文による２進表記でしょう．これはMSX-BASICのスクリーン・エディタがあってこその便利さであり，最もパターンの形をイメージしやすい表記だと思います．８×８ドットのパターンの場合は，これを利用すれば十分です．

しかし，16×16ドットのパターンを作るとなると，パターンも大きくなりますし，DATA文の読み取り方も異なり，８×８ドットのときのように見た目どおりにはいきません．さらに，パターン全体を１ドット分だけ，上下左右のいずれかへずらす(シフトする)などといったことをしようとすると，キー操作が非常に多くなってかなりめんどうです．上下方向ならば，行番号のつけ換え程度ですみますが，左右のどちらかへシフトするときはたいへんです．一方，DATA文に16進数や文字列を用いたときには，実際に表示するとどうなるか，ピンとこないという不都合があります．

そこでこの章では，スプライト・パターンのデータ作成に役立てるために，いま述べたような不都合を解消するさまざまな機能を持ったツール(道具)を作ってみることにします．

DATA文の２進表記ではめんどうだった，パターン作りの基本的な操作を中心として考えていけば，比較的便利なツールができ上がると思います．機能として持っていると便利そうなものを，いくつかあげてみましょう．

**(1) パターンの作成(点を打つ，消す)**
**(2) パターンのシフト**
**(3) パターンのローテイト**
**(4) スプライト・パターンとしての表示**
**(5) パターンの保存(データ化，数値化)**

基本的な点では，この程度の機能を備えていれば，少なくともDATA文の２進表記よりは使い勝手がいいと思います．それでは，使い方を考えながら，各機能の実際の働きなど，ツールの全体的な仕様について話を進めていきましょう．

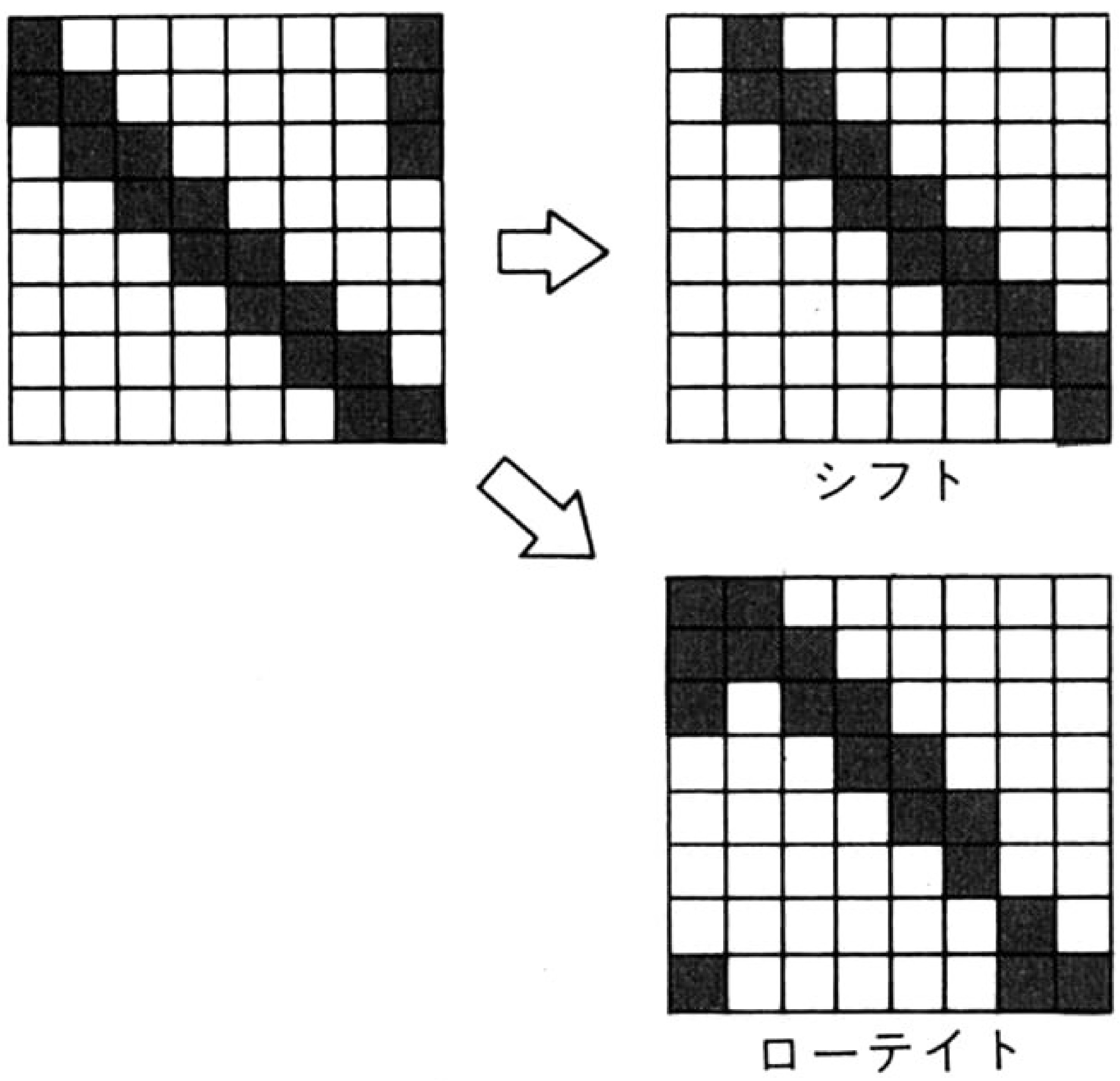
# 2 データ作りプログラムの使い方

(1)のパターンの作成はDATA文の2進表記の例でいえば，カーソルを使って0と1のキーでDATA文の2進数を書き換えるのと同じです．ツールでは，見やすさを考えてパターンを画面上に大きく表示しながら作成できるようにしますので，カーソルをプログラムで用意しなければなりません．したがって，スプライト機能を応用してカーソルを作ることにしましょう．

キーボードを見ると，0と1のキー位置が遠く，このことがデータを作る手間の1つになっていますので，操作性を考えてスペースバーを打つごとにカーソルの位置する点の色が反転するようにします．

(2)と(3)の機能は，**図6－1**にそれぞれ示したようなもので，パターンを全体的に任意の方向へ移動させる機能です．**図6－1**の例は，右方向に作用させたときのものです．

**図6－1**

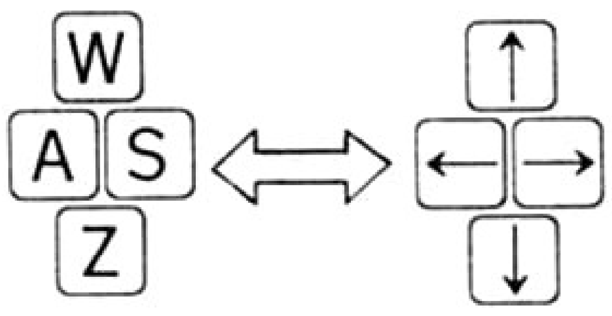



これらの機能は，当然，上下左右のすべての方向に対して動作したほうが便利でしょう．

この命令のために，キーボードの左のほうに並んでいる，W，Z，A，Sの4つのキーを上下左右に割り当てて（**図6－2**参照），その入力が小文字ならローテイ

ト，大文字——つまり，[SHIFT]＋小文字ならシフトとなるようにプログラムを組みます．

図6－2



(4)の実際の大きさで表示するという機能は，拡大表示(これはデータ作成中の状態)──→通常のスプライト・パターンの大きさで表示──→2倍表示を[SELECT]キーで循環するようにします．データは実際のスプライト・パターンとして登録され，スプライト・プレーンにPUTします．今回のツールでは，パターンの表示のサイズが変わるたびに画面も変わるようになっていますが，3つのサイズを同じ画面に表示するようにすれば，さらに便利なものとなるでしょう．

(5)のパターンの保存は，はじめはカセットやディスクにセーブできる機能にするつもりでしたが，16進数でデータを表示してしまえば，たいした量ではないので，画面に表示するだけにしました．書きうつすのもそれほど手間をとらないでしょう．行番号と〝DATA〟という文字列を合わせて表示しますので，BASICの命令文として取り込むこともできます．

望みのパターンができたら[RETURN]キーを押してください．Retry？というツールの質問に〝n〟で答えて，〝Ok〟の表示が出たらカーソルを移動させ，行番号のところでふたたび[RETURN]キーをたたきます(この際,ツールのプログラムを「new」してからのほうが,作ったデータにプログラムを加えるときに便利です)．なお，〝y〟(小文字のみ)と答えた場合には，もとの画面にもどってデータ作成作業を続けることができます．

以上の機能を用いて作ったパターンのデータは，ツールのプログラムの内部では，各点を配列の1つ1つの要素に割り当てています．配列は2次元にしてデータの参照を簡単にしており，点のあるところの値を－1，ないところの値を0として示しています．

## リスト6-1

```
1000 '
1010 '  TOOL FOR SPRITE PATTERN
```

```
1020 '
1030 GOTO 2350
1040 '
1050 SPRITE$(0)=STRING$(2,255)+STRING$(4
,195)+STRING$(2,255):RETURN
1060 '
1070 ' DISPLAY ALL DOT PATTERN
1080 '
1090 SCREEN 2,0:CLS:GOSUB 1050
1100 FOR J=0 TO S:FOR I=0 TO S
1110  A=I*8+48:B=J*8+16
1120  IF P(I,J) THEN GC=9 ELSE GC=1
1130  LINE(A,B)-STEP(7,7),GC,BF
1140 NEXT:NEXT
1150 OPEN "GRP:" FOR OUTPUT AS #1
1160 PRESET(30,0)
1170 PRINT #1,">THE TOOL FOR SPRITE<"
1180 CLOSE #1
1190 RETURN
1200 '
1210 ' ROTATE RIGHT
1220 '
1230 FOR J=0 TO S:FOR I=S TO 1 STEP -1
1240  SWAP P(I-1,J),P(I,J)
1250 NEXT:NEXT
1260 RETURN
1270 '
1280 ' SHIFT RIGHT
1290 '
1300 GOSUB 1230
1310 FOR I=0 TO S:P(0,I)=0:NEXT
1320 RETURN
1330 '
1340 ' ROTATE LEFT
1350 '
1360 FOR J=0 TO S:FOR I=1 TO S
1370  SWAP P(I-1,J),P(I,J)
1380 NEXT:NEXT
1390 RETURN
1400 '
1410 ' SHIFT LEFT
1420 '
1430 GOSUB 1360
1440 FOR I=0 TO S:P(S,I)=0:NEXT
```

```
1450 RETURN
1460 '
1470 ' ROTATE ABOVE
1480 '
1490 FOR I=0 TO S:FOR J=1 TO S
1500  SWAP P(I,J-1),P(I,J)
1510 NEXT:NEXT
1520 RETURN
1530 '
1540 ' SHIFT ABOVE
1550 '
1560 GOSUB 1490
1570 FOR I=0 TO S:P(I,S)=0:NEXT
1580 RETURN
1590 '
1600 ' ROTATE BELOW
1610 '
1620 FOR I=0 TO S:FOR J=S TO 1 STEP -1
1630  SWAP P(I,J-1),P(I,J)
1640 NEXT:NEXT
1650 RETURN
1660 '
1670 ' SHIFT BELOW
1680 '
1690 GOSUB 1620
1700 FOR I=0 TO S:P(I,0)=0:NEXT
1710 RETURN
1720 '
1730 ' INITIALIZE
1740 '
1750 COLOR 9,0,0:SCREEN 0:O$=""
1760 FOR I=1 TO 9
1770  READ D:O$=O$+CHR$(D)
1780 NEXT
1790 PRINT "PATTERN SIZE:"
1800 INPUT "1)8*8 OR 2)16*16 ";S
1810 IF S=1 THEN S=7 ELSE S=15
1820 DIM P(S,S):X=0:Y=0:SCREEN 2
1830 COLOR 15,4,4
1840 RETURN
1850 '
1860 ' MAIN LOOP
1870 '
1880 GOSUB 1090
```

```
1890 A=X*8+48:B=Y*8+16
1900 PUT SPRITE 0,(A,B),2
1910 C=ASC(INPUT$(1))
1920 IF C=13 THEN 2170
1930  F=INSTR(O$,CHR$(C))
1940  IF F=0 THEN 1980
1950  ON F GOSUB 1230,1300,1360,1430,
          2690,1490,1560,1620,1690
1960  GOSUB 1090:GOTO 2030
1970 '
1980  IF C=28 AND X<S THEN X=X+1
1990  IF C=29 AND X>0 THEN X=X-1
2000  IF C=30 AND Y>0 THEN Y=Y-1
2010  IF C=31 AND Y<S THEN Y=Y+1
2020  IF C=32 THEN 2070
2030  A=X*8+48:B=Y*8+16
2040  PUT SPRITE 0,(A,B),2
2050  GOTO 2120
2060 '
2070  P(X,Y)=P(X,Y) XOR -1
2080  A=X*8+48:B=Y*8+16
2090  IF P(X,Y) THEN GC=9 ELSE GC=1
2100  LINE(A,B)-STEP(7,7),GC,BF
2110  '
2120  C=ASC(INPUT$(1))
2130 GOTO 1920
2140 '
2150 ' DISPLAY ALL DOT DATA
2160 '
2170 N=10000:V=0:W=0:LM=-1
2180 GOSUB 2520
2190 SCREEN 0:CLS
2200 IF S=7 THEN 2280
2210 PRINT L$:N=N+10:W=8
2220 GOSUB 2520
2230 PRINT L$:N=N+10:V=8:W=0
2240 GOSUB 2520
2250 PRINT L$:N=N+10:W=8
2260 GOSUB 2520
2270 '
2280 PRINT L$:PRINT:PRINT
2290 INPUT "RETRY ";L$
2300 IF L$="y" THEN 1880
2310 RETURN
```

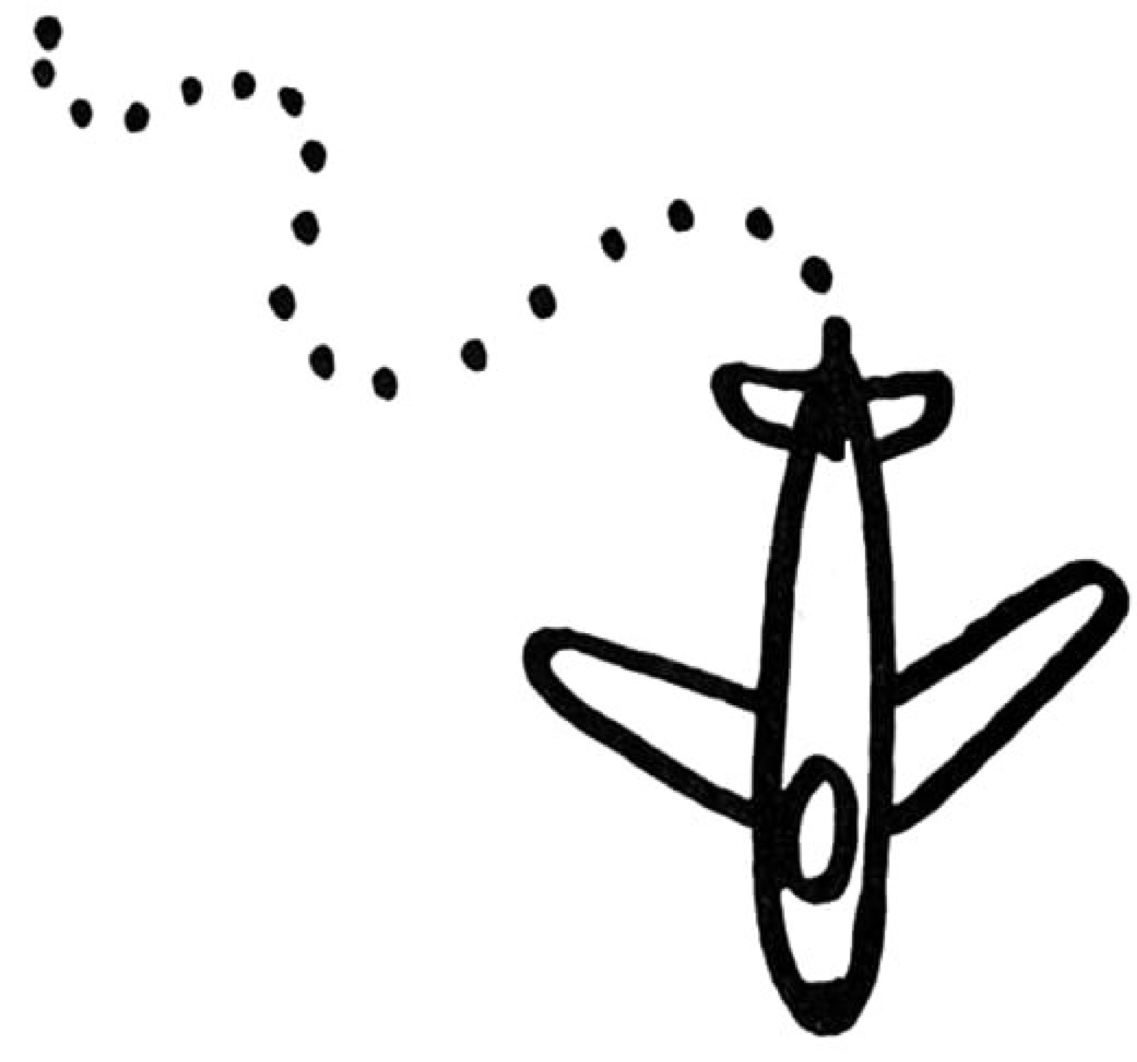
```
2320 '
2330 '  MAIN ENTRY
2340 '
2350 GOSUB 1750
2360 GOSUB 1880
2370 END
2380 '
2390 DATA 115,83,97,65,24
2400 DATA 119,87,122,90
2410 '
2420 '  CONVERT DOT IMAGE INTO SPRITE
2430 '
2440 FOR I=0 TO 7
2450  IF P(V+I,W+J) THEN D=D+2^(7-I)
2460 NEXT
2470 IF LM THEN L$=L$+"&H"+HEX$(D)+","
2480 RETURN
2490 '
2500 '  CONVERT PATTERN INTO DATA
2510 '
2520 L$=RIGHT$(STR$(N),LEN(STR$(N))-1)
2530 L$=L$+" DATA "
2540 FOR J=0 TO 7:D=0
2550  GOSUB 2440
2560 NEXT
2570 L$=LEFT$(L$,LEN(L$)-1)
2580 RETURN
2590 '
2600 '  MAKE DATA
2610 '
2620 FOR J=0 TO 7:D=0
2630  GOSUB 2440:S$=S$+CHR$(D)
2640 NEXT
2650 RETURN
2660 '
2670 '  CHECK SPRITE PATTERN
2680 '
2690 S$="":V=0:W=0:LM=0
2700 GOSUB 2620
2710 IF S=7 THEN SS=0 ELSE SS=2
2720 IF S=7 THEN 2770
2730 W=8:GOSUB 2620
2740 V=8:W=0:GOSUB 2620
2750 W=8:GOSUB 2620
```

```
2760 '
2770 CLS:PUT SPRITE 0,(0,209)
2780 '
2790 SCREEN ,SS:SPRITE$(1)=S$
2800 PUT SPRITE 1,(128-S,96-S)
2810 C=ASC(INPUT$(1))
2820 IF C<>24 THEN 2810
2830 SCREEN ,SS+1:SPRITE$(1)=S$
2840 PUT SPRITE 1,(128-S*2,96-S*2)
2850 C=ASC(INPUT$(1))
2860 IF C<>24 THEN 2850
2870 PUT SPRITE 1,(0,209)
2880 RETURN
```

このツールによって，みなさんのパターン作りの手間が，少しでも軽減することを期待します．

いままでお話してきたツールは，単色のパターンを作ることを目的としていましたが，複数色のパターンにもこのツールは流用できます．

1つのパターンに複数の色をつける場合，5章で述べたように同時に4つの色までしか使えません．したがって，多色のパターンは使用する色別にパターンを分解して，それぞれのパターンをこのツールで1つずつ作っていきます．そして，できたデータを，使用したいプログラム中で利用するときに，各パターンに色を与えてPUTすればよいのです．





GAME CHARACTER

# 7

# スプライト・オン・ゲーム



これまでは，パターンの表示・移動をおもに扱ってきました．そこで，この章ではすぐにゲーム作りに役立つ，スプライトのもう一つのテクニックを紹介しましょう．

# 1 割り込み機能

スプライトの特徴は，スプライト・パターンが重なったり，すれ違ったりしても何も起きないことでした．ですから，インベーダーゲームのビームとインベーダーをスプライト・パターンで作った場合，インベーダーにビームが当たっても単にすれ違うだけです．

ところが，ゲームとしてはこれでは困ります．ビームが当たったかどうかをチェックし，インベーダーを消すようにしなければ，ゲームとしては不都合です．

ビームがインベーダーに当たったかどうかのチェックは，ここまで述べてきたテクニックの中では，IF文などの条件文を使うことしか思い当たりません．しかし実際には，IF文ではとても遅くてゲームになりません．なぜなら，ビームが移動するたびに，そこにインベーダーがあるかどうかをいちいち調べなければならないからです．

そこで，便利なのがスプライトの**割り込み機能**です．このスプライトの割り込み機能は，画面に表示されているスプライト・パターンどうしが重なったとき，割り込みが発生するものです．

「**割り込み**」とは，あることをしているときに，予告なしに他の作業にうつることをいいます．そのときを「割り込みがかかった」といい，割り込みがかかったときに行なわれる作業を「**割り込み処理**」といいます．

たとえば，画面に長いプログラムのリストをとったとき，途中で[STOP]キーを押すと画面が一時的に止まります．つまり，[STOP]キーを押すことで「割り込み」がかかり，画面を一時停止させる「割り込み処理」をしているのです．

スプライトの場合，その割り込み処理は私たちが自由に指定することができます．つまり，インベーダーとビームが当たったとき「すぐにインベーダーを消して，点を加算する」という処理を指定できるのです．

この場合割り込み処理はプログラムの一部で指定します．そこで，プログラムのどの部分が，「割り込み処理」であるのかを指定する命令が，ON SPRITE

GOSUB文です.

リスト7-1

```
100 COLOR 15,1,1:SCREEN 2,1:CLS
110 GOSUB 280
120 X=100:Y=90:A=140:B=90
130 DX=-2:DY=-3:DA=3:DB=2
140 C0=5:C1=9
150 '
160 ON SPRITE GOSUB 470:SPRITE ON
170 '
180 PUT SPRITE 0,(X,Y),C0
190 PUT SPRITE 1,(A,B),C1
200 X=X+DX:Y=Y+DY
210 IF X<5 OR X>236 THEN
     X=X-DX:DX=-1*DX:SPRITE ON
220 IF Y<0 OR Y>175 THEN
     Y=Y-DY:DY=-1*DY:SPRITE ON
230 A=A+DA:B=B+DB
240 IF A<5 OR A>236 THEN
     A=A-DA:DA=-1*DA:SPRITE ON
250 IF B<0 OR B>175 THEN
     B=B-DB:DB=-1*DB:SPRITE ON
260 GOTO 180
270 '
280 FOR I=1 TO 8
290  READ P$,Q$
300  P=VAL("&B"+P$):Q=VAL("&B"+Q$)
310  S0$=S0$+CHR$(P)
320  S1$=S1$+CHR$(Q)
330 NEXT
340 SPRITE$(0)=S0$
350 SPRITE$(1)=S1$
360 RETURN
370 '
380 DATA 00010000,00111000
390 DATA 00111000,01000100
400 DATA 01010100,01000100
410 DATA 10010010,00111000
420 DATA 00111000,00010000
430 DATA 01000100,11111110
440 DATA 01000100,00010000
```

```
450 DATA 00111000,00010000
460 '
470 SPRITE OFF
480 SWAP C0,C1
490 DX=-1*DX:DY=-1*DY
500 DA=-1*DA:DB=-1*DB
510 RETURN
```

リストに見られるように，行番号160のON SPRITE GOSUB文のあとには，数字があります．この数字は行番号で，その行番号から「割り込み処理」であることを指示しているのです．このプログラムでは割り込みがかかったとき，2つのパターンの進む向きを変え，色を入れ換えています．「ON SPRITE GOSUB～」という命令文からもわかるように，「割り込み処理」はサブルーチンの形で用意します．ですから，割り込みがかかってその処理を終えると，RETURN文で割り込みの起きた時点に戻って，プログラムの続きを実行します．当然，RETURN文で行番号を指定すれば，指定場所に戻すこともできます．

ここで注意してほしいのは，ON SPRITE GOSUB文がプログラムのどの行番号から「割り込み処理」であるかを指定しているだけだということです．「割り込み処理」が実行されるのは，あくまでスプライト・パターンが画面上で重なったときです．そこで，ゲームに活かす場合には，ON SPRITE GOSUB文はゲームスタート前に用意しておきます．

次に，SPRITE ONという命令があります．これが，「スプライト・パターンが重なったら割り込み処理をしなさい」という意味の命令文です．これを実行しておかないと，画面上でスプライト・パターンが重なっても割り込みがかかりません．

ところで，割り込み処理中にさらに割り込みがかかったらどうなるのでしょう．

MSXは割り込み処理をしているときに割り込みがかかっても，ひととおりの処理を終えるまでは，他の処理へ分岐することはありません．割り込み処理中に起きた割り込みは，その処理を終えてRETURN文で処理ルーチンを抜けたあとに，ふたたび割り込みがかかったものと見なされます．

このような状態は，SPRITE ONでありながら例外的な状態です．ここでは，「割り込み処理はなされないが，割り込みがかかったことはわかっている」わけです．これは，じつはSPRITE STOPという命令が実行されているのと同じ状態で，MSXは割り込み処理中には自動的にこの状態になるようにできています．

一方，割り込みがダブる場合にかぎらず，スプライト・パターンが重なっても割り込みがかかってほしくないこともあります.こういうときに使う命令がSPRITE OFFです.

たとえば，インベーダーの出したビームが，砲台に当たったときのことを考えてみましょう．この砲台は，割り込み処理中に消されます．しかし，砲台を消す前に次のビームが当たってしまったら，その割り込みは無効にしなければなりません．なぜなら，前の割り込みが有効のままだと，次の砲台はビームが当たっていないのに，当たったものと見なされてしまうことがあるからです．この場合，割り込み処理中に起きたもう1つの割り込みは，不必要なものなのでSPRITE OFF文で無効にしておくのです.

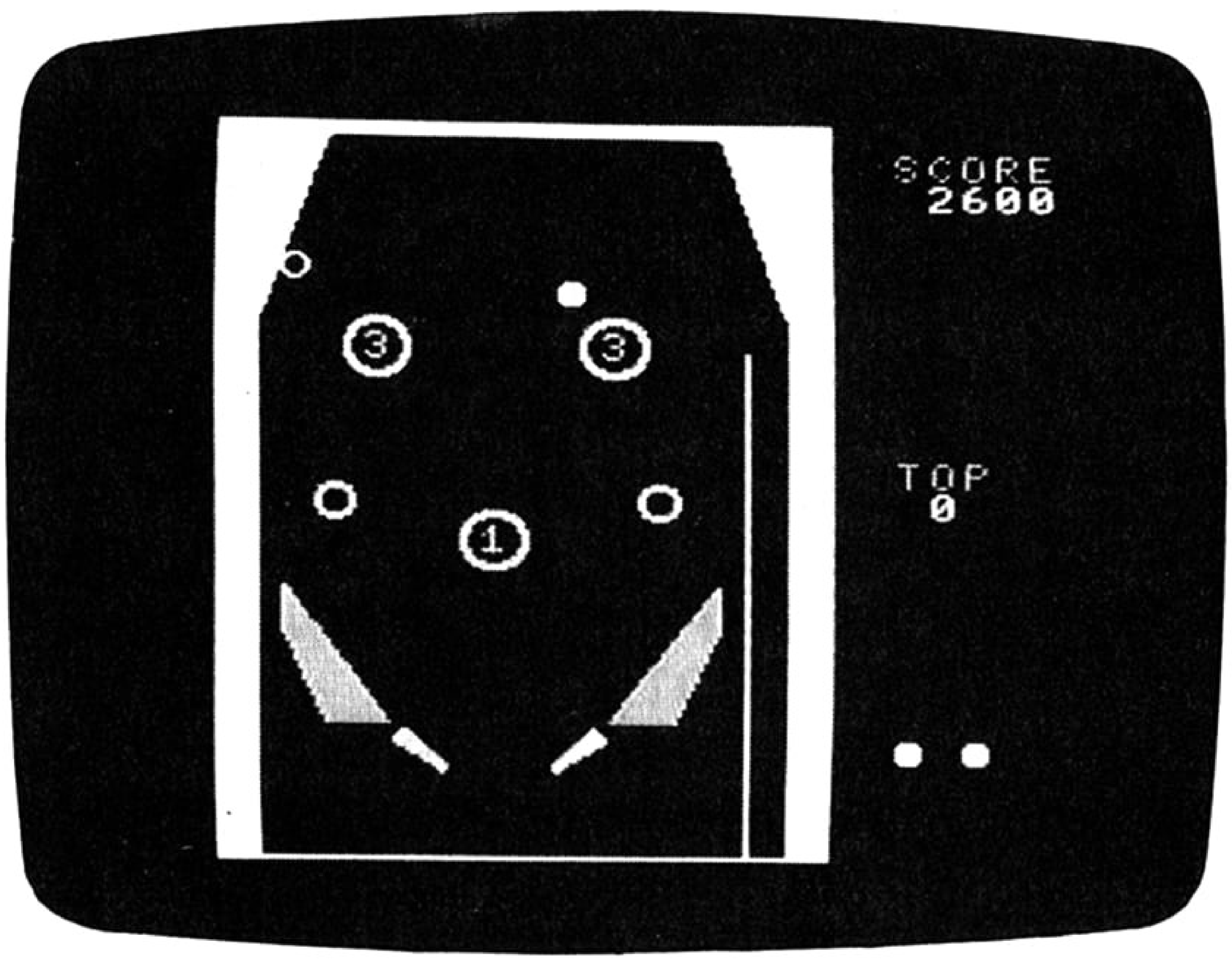
**リスト7－1**のプログラムの中で，割り込み処理ルーチンのはじめに(行番号470)にSPRITE OFF状態にしてあるのは，割り込みを回避しているのです．当然，SPRITE OFFにすると，割り込み処理もまったくきかなくなりますから，SPRITE ONで，ふたたび有効にしなければなりません.

こうした割り込み機能の細かいテクニックは，**リスト7－1**，次にあげる**7－2**でしっかり学びとってください.

## 2 ピンボールで遊ぼう

前述したスプライトの割り込み機能を用いて，ゲームを作ってみましょう.

**画面7－1**



リスト7-2

```
1000 GOTO 2200
1010 '
1020 STRIG(0) OFF:F=-1
1030 PUT SPRITE 10,(94,157),7,7
1040 PUT SPRITE 11,(61,157),7,8
1050 FOR W=0 TO 20:NEXT
1060 PUT SPRITE 10,(94,157),7,5
1070 PUT SPRITE 11,(61,157),7,6
1080 F=0:STRIG(0) ON:RETURN
1090 '
1100 SPRITE OFF
1110 SOUND 7,252:SOUND 0,32
1120 SOUND 1,0:SOUND 2,52
1130 SOUND 3,0:SOUND 8,16
1140 SOUND 9,16:SOUND 11,0
1150 SOUND 12,40:SOUND 13,0
1160 '
1170 IF Y>110 THEN 1330
1180 IF Y>74 THEN LP=LP+50:GOTO 1250
1190 LP=LP+300:IF X>77 THEN 1230
1200 ' no.2
1210 A=X-50:B=Y-50:GOTO 1430
1220 ' no.1
1230 A=X-104:B=Y-50:GOTO 1430
1240 ' no.4
1250 IF X<56 THEN 1290
1260 IF X>98 THEN 1310
1270 A=X-77:B=Y-100:LP=LP+50:GOTO 1430
1280 ' no.5
1290 A=X-40:B=Y-90:GOTO 1430
1300 ' no.3
1310 A=X-114:B=Y-90:GOTO 1430
1320 ' flipper
1330 IF X<57 THEN 1390
1340 IF X>114 THEN 1410
1350 IF F THEN P=-45 ELSE P=-ABS(P)+5
1360 IF X>77 THEN DX=-.8 ELSE DX=.8
1370 GOTO 1500
1380 ' no.8, 9
1390 P=-ABS(P):DX=1:GOTO 1500
1400 ' no.6,'7
1410 P=-ABS(P):DX=-1:GOTO 1500
```

```
1420 '
1430 IF ABS(P)<9 THEN Y=Y-SGN(P)*2
1440 X=X-DX*2:Y=Y-SGN(P)*DY(ABS(P))*2
1450 PUT SPRITE 0,(X,Y)
1460 IF ABS(A)>.05 THEN AT=ABS(ATN(B/A))
    ELSE DX=.2:P=SGN(B)*20:GOTO 1500
1470 DX=SGN(A)*COS(AT)*1.8
1480 P=SGN(B)*SIN(AT)*18
1490 '
1500 X=X+DX*2:Y=Y+SGN(P)*DY(ABS(P))
1510 PUT SPRITE 0,(X,Y):P=P+1
1520 LINE(176,16)-STEP(48,8),1,BF
1530 STRIG(0) OFF:PRESET(176,16)
1540 PRINT #1,LP:STRIG(0) ON
1550 SPRITE ON:RETURN
1560 ' sprite pattern maked
1570 FOR J=0 TO 8:S$=""
1580  FOR I=1 TO 32:READ D$
1590   S$=S$+CHR$(VAL("&H"+D$))
1600  NEXT
1610  SPRITE$(J)=S$
1620 NEXT
1630 RETURN
1640 '
1650 COLOR 15,1,1:SCREEN 2,2:CLS
1660 DIM DY(60),OX(65),OY(65)
1670 PI=3.14159
1680 FOR I=0 TO 60:DY(I)=(I+1)/10:NEXT
1690 FOR I=6 TO 71
1700  OX(I-6)=60*COS(-PI*I/90)+81
1710  OY(I-6)=60*SIN(-PI*I/90)+60
1720 NEXT
1730 GOSUB 1570:' read sprite data
1740 ON STRIG GOSUB 1020
1750 ON SPRITE GOSUB 1100
1760 '
1770 LINE(20,0)-(29,191),,BF
1780 LINE(151,0)-(160,191),,BF
1790 LINE(20,191)-(160,191)
1800 LINE(141,60)-(141,191)
1810 LINE(30,50)-(45,3)
1820 LINE-STEP(90,0):LINE-(150,50)
1830 PAINT(35,5)
1840 LINE(134,121)-(110,156),3
```

```
1850 LINE-STEP(15,0),3
1860 LINE-(135,132),3
1870 LINE-(135,121),3
1880 LINE(36,121)-(60,156),3
1890 LINE-STEP(-15,0),3
1900 LINE-(35,132),3
1910 LINE-(35,121),3
1920 PAINT(130,140),3
1930 PAINT(40,140),3
1940 CIRCLE(38,38),3
1950 '
1960 PUT SPRITE 1,(104,50),9,1
1970 PUT SPRITE 2,(50,50),9,1
1980 PUT SPRITE 3,(114,90),9,2
1990 PUT SPRITE 4,(77,100),9,1
2000 PUT SPRITE 5,(40,90),9,2
2010 PUT SPRITE 6,(120,120),3,3
2020 PUT SPRITE 7,(106,140),3,3
2030 PUT SPRITE 8,(35,120),3,4
2040 PUT SPRITE 9,(49,140),3,4
2050 PUT SPRITE 10,(94,157),7,5
2060 PUT SPRITE 11,(61,157),7,6
2070 '
2080 OPEN "GRP:" FOR OUTPUT AS #1
2090 COLOR 3:PRESET(176,8)
2100 PRINT #1,"SCORE"
2110 PRESET(176,88):PRINT #1,"TOP"
2120 COLOR 9
2130 PRESET(109,55):PRINT #1,"3"
2140 PRESET(55,55):PRINT #1,"3"
2150 PRESET(82,105):PRINT #1,"1"
2160 RETURN
2170 '
2180 ' main entry
2190 '
2200 GOSUB 1650:' initialize
2210 '
2220 W$=INKEY$
2230 IF W$<>CHR$(13) THEN 2220
2240 LP=0:LB=5:STRIG(0) ON:SPRITE ON
2250 LINE(176.16)-STEP(40,8),1,BF
2260 COLOR 15:PRESET(176,16)
2270 PRINT #1.LP:PRESET(176,160)
2280 PRINT #1."●●●●"
```

```
2290 LINE(176.96)-STEP(40,8),1,BF
2300 PRESET(176.96):PRINT #1,HI
2310 '
2320 IF LB=0 THEN 2540
2330 ES=160+LB*16
2340 LINE(ES.160)-STEP(32,8),1,BF
2350 PUT SPRITE 0.(139.175)
2360 FOR I=175 TO 48 STEP -1
2370  PUT SPRITE 0.(139.I)
2380 NEXT
2390 FOR I=0 TO 65
2400  FOR M=0 TO 10:NEXT
2410  PUT SPRITE 0.(OX(I),OY(I))
2420 NEXT
2430 X=OX(65):Y=OY(65)
2440 DX=1.5+INT(RND(-TIME)*7+1)/10-.3
2450 P=-12+INT(RND(-TIME)*7+1)-3
2460 '
2470 IF X<25 OR X>130 THEN DX=-1*DX
2480 IF Y<3 THEN P=ABS(P)
2490 IF Y>184 THEN LB=LB-1:GOTO 2320
2500 PUT SPRITE 0.(X.Y)
2510 X=X+DX:Y=Y+SGN(P)*DY(ABS(P))
2520 P=P+1:GOTO 2470
2530 '
2540 SPRITE OFF:STRIG(0) OFF
2550 IF LP>HI THEN HI=LP
2560 LINE(176.96)-STEP(48,8),1,BF
2570 PRESET(176.96):PRINT #1,HI
2580 GOTO 2220
2590 '
2600 DATA 00.00.00,00.00,03,07,07
2610 DATA 07.07.03,00,00,00,00,00
2620 DATA 00.00,00,00,00,C0,E0,E0
2630 DATA E0.E0,C0,00,00,00,00,00
2640 '
2650 DATA 07,1F,3C,70,60,E0,C0,C0
2660 DATA C0,C0,E0,60,70,3C,1F,07
2670 DATA E0,F8,3C,0E,06,07,03,03
2680 DATA 03,03,07,06,0E,3C,F8,E0
2690 '
2700 DATA 00,00,00,03,0F,0C,18,18
2710 DATA 18,18,0C,0F,03,00,00,00
2720 DATA 00,00,00,C0,F0,30,18,18
```

```
2730 DATA 18,18,30,F0,C0,00,00,00
2740 '
2750 DATA 00,00,00,00,00,00,00,00
2760 DATA 00,00,01,01,03,07,07,0F
2770 DATA 03,07,07,0F,1F,1F,3F,7F
2780 DATA 7F,FF,FF,F0,E0,C0,80,00
2790 '
2800 DATA C0,E0,E0,F0,F8,F8,FC,FE
2810 DATA FE,FF,FF,0F,07,03,01,00
2820 DATA 00,00,00,00,00,00,00,00
2830 DATA 00,00,80,80,C0,E0,E0,F0
2840 '
2850 DATA 00,00,00,00,00,01,03,07
2860 DATA 0F,1E,1C,08,00,00,00,00
2870 DATA 0C,1E,3F,7F,FE,F8,F0,C0
2880 DATA 80,00,00,00,00,00,00,00
2890 '
2900 DATA 30,78,FC,FE,7F,1F,0F,03
2910 DATA 01,00,00,00,00,00,00,00
2920 DATA 00,00,00,00,00,80,C0,E0
2930 DATA F0,78,38,10,00,00,00,00
2940 '
2950 DATA FF,FF,FF,1F,00,00,00,00
2960 DATA 00,00,00,00,00,00,00,00
2970 DATA FE,FF,FF,FF,FE,00,00,00
2980 DATA 00,00,00,00,00,00,00,00
2990 '
3000 DATA 7F,FF,FF,FF,7F,00,00,00
3010 DATA 00,00,00,00,00,00,00,00
3020 DATA FF,FF,FF,F8,00,00,00,00
3030 DATA 00,00,00,00,00,00,00,00
```

**リスト 7－2** のゲームは，RUNさせると，しばらくたってからピンボールの盤が表示されます．ゲームはリターンキーを押すとスタートし，フリッパーはスペースバーを押すごとに動くようになっています．ボールは全部で5つあり，残りは画面右下に表示されています．

ゲームが終了すると，ハイスコアが書き換えられます．もう1度挑戦したいときは，リターンキーを押してゲーム再開です．

さて，あなたの得点は何点ですか？

## ゲームキャラクタ操縦法　MSXポケットバンク⑤

1984年4月5日　初版発行　定価480円

著　者　横溝和宏

発行者　塚本慶一郎

発行所　株式会社アスキー

〒107 港区南青山5－11－5住友南青山ビル5F

振　替　東京4-161144

電　話　03－486－7111（代表）





編集担当　八木淳一

表紙担当　郷　啓子

印刷　図書印刷株式会社

ISBN4-87148-737-7 C0055 ¥480E

MSX
POCKET BANK

MSXポケットバンク5

# ゲームキャラクタ操縦法

## CONTENTS





ISBN4-87148-737-7 C0055 ¥480E

定価480円